IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: Eric M. Dowling	§	Confirmation No. 7948
Serial No.: 10/074,779	§	Group Art Unit: 2183
Filed: February 13, 2002	§	Examiner: Huisman, David J.
For: Program Controlled Embedded-DRAM-DSP Having Improved Instruction Set Architecture	§	Atty Docket: MICS:0171-2/MAN/LIU 1998-0010.02

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313

CERTIFICATE OF TRANSMISSION OR MAILING
37 C.F.R. 1.8

I hereby certify that this correspondence is being transmitted by facsimile to the United States Patent and Trademark Office in accordance with 37 C.F.R. 1.6(d) or is being deposited with the U.S. Postal Service with sufficient postage as First Class Mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date below:

October 22, 2008	/Robert A. Manware/
Date	Robert. A. Manware

APPEAL BRIEF PURSUANT TO 37 C.F.R. §§ 41.31 AND 41.37

This Appeal Brief is being filed in furtherance to the Notice of Appeal electronically filed and received by the Patent Office on July 28, 2008. A Pre-Appeal Brief Request for Review was filed concurrently with the Notice of Appeal. A Panel Decision indicating that the present application should proceed to the Board of Appeals was mailed on September 22, 2008.

The Commissioner is authorized to charge the requisite fee of $540.00, and any additional fees which may be necessary to advance prosecution of the present application, to Deposit Account No. 06-1315, Order No. MICS:0171-2/MAN.

I. REAL PARTY IN INTEREST

The real party in interest is Micron Technology, Inc., the Assignee of the above referenced application by virtue of the Assignment recorded at reel 9009, frame 0502, from the inventor, Eric M. Dowling, to Micron Technology, Inc., dated January 24, 1998. Accordingly, Micron Technology, Inc. will be directly affected by the Board's decision in the pending appeal.

II. RELATED APPEALS AND INTERFERENCES

The Appellant is unaware of any other appeals or interferences related to this Appeal. The undersigned is the Appellant's legal representative in this Appeal.

III. STATUS OF CLAIMS

The present application was originally filed with claims 1-49. Claim 50 was added by amendment in response to the Office Action mailed December 9, 2004. *See* Response to Office Action mailed 12/9/2004, page 18. Claims 15 and 33 were subsequently canceled in the response to the Office Action mailed July 25, 2005. *See* Response to Office Action mailed 7/25/2005, pages 6, 10. Further, claim 48 was canceled and claim 51 was added by amendment in the previously filed Response to the Office Action mailed August 21, 2007. *See* Response to Office Action mailed 8/21/2007, page 18; *see also* Response to Notice Non-Compliant Amendment mailed 2/6/2008, page 18. Accordingly, claims 1-14, 16-32, 34-47, and 49-51 are currently under final rejection and, thus, are the subject of the present Appeal.

IV. STATUS OF AMENDMENTS

The instant claims have not been amended subsequent to the Final Office Action mailed May 28, 2008. Consequently, there are no outstanding amendments to be considered by the Board.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The present invention relates generally to the field of computing micro-systems architecture and, more particularly, to memory access, memory hierarchy, and memory control strategies for use in embedded-DRAM (dynamic random access memory) digital signal processors (DSP's) and media processors. *See, e.g.,* Application, page 1, lines 7-11.

The Application contains nine independent claims, namely, claims 1, 13, 16, 23, 28, 45, 46, 49, and 50, all of which are the subject of this Appeal. The subject matter of these claims is summarized below.

With regard to the aspect of the invention set forth in independent claim 1, discussions of the recited features of claim 1 can be found at least in the below cited locations of the specification and drawings. By way of example, an embodiment in accordance with claim 1 relates to an embedded-DRAM (dynamic random access memory) processor. *See, e.g., id.* at page 4, lines 19-21; FIG. 1. The embedded-DRAM processor includes an embedded DRAM array (*e.g.*, 102, 534) having a plurality of random access memory cells arranged in rows and columns. *See, e.g., id.* at page 6, lines 22-26; page 7, lines 7-10; page 8, lines 7-9 and 18-20; page 9, lines 14-20 and 26-29; page 10, lines 15-18; page 13, lines 16-18; page 22, lines 7-23; page 36, lines 12-22; FIGS. 1, 4-5. The embedded-DRAM processor further includes a row address register (*e.g.*, 302) that holds a pointer that points to a row (*e.g.*, 208) of the DRAM array (*e.g.*, 102, 534). *See, e.g., id.* at page 7, lines 7-16; page 8, lines 18-28; page 26, lines 5-24; page 31, lines 13-17; page 33, lines 12-19; FIGS. 1, 3-4. The embedded-DRAM processor further includes one or more sets of registers (*e.g.*, registers 0-k in register file 112), wherein each of the sets of registers is capable of loading or storing an entire row (*e.g.*, 208) of the DRAM array (*e.g.*, 102, 534) in response to a single latch signal. *See, e.g., id.* at page 6, lines 25-29; page 7, lines 13-16 and 24-26; page 8, lines 7-22; page 10, lines 17-21; page 28, line 21 – page 30, line 12; FIGS. 2-3. The embedded-DRAM

processor further includes an instruction set having at least one command to perform an arithmetic operation on said row address register (*e.g.*, 302). *See, e.g., id.* at page 8, line 18 – page 9, line 13; page 10, lines 22-28; page 11, lines 1-25; page 24, lines 18-21; FIG. 3. The instruction set further includes a command to precharge (activate) the row pointed to by said row address register (*e.g.*, 302). *See, e.g., id.* at page 8, line 18 – page 10, line 14; page 14, line 10 – page 15, line 7; page 26, line 24 – page 27, line 6; page 27, lines 17-29; page 30, line 24 – page 31, line 12; page 32, line 24 – page 33, line 19; FIG. 3. The instruction set further includes a command to deactivate said row pointed to by said row address register (*e.g.*, 302) after it had been precharged by the command to precharge. *See, e.g., id.* at page 8, line 18 – page 9, line 13; page 10, lines 22-30; page 21, lines 1-9; page 26; lines 5-23; page 33, lines 6-19; FIG. 3. The instruction additionally includes a command to load a plurality of words of the precharged row into designated sets of data registers (*e.g.*, registers 0-k in register file 112). *See, e.g., id.* at page 7, lines 7-26; page 8, line 18 – page 9, line 13; page 10; lines 15-30; page 11, lines 1-14; page 15, line 29 – page 16, line 11; page 31, lines 13-20; page 32, line 24 – page 33, line 5; FIGS. 2-3. Finally, the instruction set includes a command to load selected columns of the precharged row into designated sets of data registers (*e.g.*, registers 0-k in register file 112), the selection being based on bits in a mask (*e.g.*, 108), and wherein all of the selected columns of the precharged row are loaded into the designated sets of data registers in a single operation. *See, e.g., id.* at page 8, line 18 – page 9, line 13; page 10, lines 17-21; page 28, line 21 – page 30, line 12; FIGS. 1-3.

Next, with regard to the aspect of the invention set forth in independent claim 13, discussions of the recited features of claim 13 can be found at least in the below cited locations of the specification and drawings. By way of example, an embodiment in accordance with claim 13 relates to an embedded-DRAM processor. *See, e.g., id.* at page 4, lines 19-21; FIG. 1. The embedded-DRAM processor includes an embedded DRAM array (*e.g.*, 102, 534) having a plurality of random access memory cells arranged in rows and columns. *See, e.g., id.* at page 6, lines 22-26; page 7, lines 7-10; page 8, lines 7-9 and

18-20; page 9, lines 14-20 and 26-29; page 10, lines 15-18; page 13, lines 16-18; page 22, lines 7-23; page 36, lines 12-22; FIGS. 1, 4-5. The embedded-DRAM processor further includes a row address register (*e.g.*, 302) that holds a pointer that points to a row (*e.g.*, 208) of the DRAM array (*e.g.*, 102, 534). *See, e.g., id.* at page 7, lines 7-16; page 8, lines 18-28; page 26, lines 5-24; page 31, lines 13-17; page 33, lines 12-19; FIGS. 1, 3-4. The embedded-DRAM processor further includes one or more sets of data registers (*e.g.*, registers 0-k in register file 112), each of said sets of data registers capable of loading or storing an entire row (*e.g.*, 208) of the DRAM array (*e.g.*, 102, 534) in response to a single latch signal. *See, e.g., id.* at page 6, lines 25-29; page 7, lines 13-16 and 24-26; page 8, lines 7-22; page 10, lines 17-21; page 28, line 21 – page 30, line 12; FIGS. 2-3. The embedded-DRAM processor further includes a bit mask (*e.g.*, 108) to select one or more data locations within at least one of said register sets (*e.g.*, registers 0-k in register file 112). *See, e.g., id.* at page 6, line 30 – page 7, line 6; page 7, line 16 – page 8, line 6; page 8, line 18 – page 9, line 13; page 16, lines 5-11; page 22, line 8 – page 23, line 2; page 26, lines 19-23; page 26, line 24 – page 27, line 6; page 28, lines 4-20; page 29, lines 3-11; FIGS. 1-2. The embedded-DRAM processor further includes an instruction set having a command to perform arithmetic operations on said row address register (*e.g.*, 302). *See, e.g., id.* at page 8, line 18 – page 9, line 13; page 10, lines 22-28; page 11, lines 1-25; page 24, lines 18-21; FIG. 3. The instruction set additionally includes a command to precharge (activate) the row pointed to by said row address register (*e.g.*, 302), wherein the command to precharge is executed to precharge the row prior to the command to load so that at the time the command to load is issued, the command to load can execute without the need to wait for the row to precharge. *See, e.g., id.* at page 8, line 18 – page 10, line 14; page 14, line 10 – page 15, line 7; page 15, line 29 – page 16, line 22; page 26, line 24 – page 27, line 6; page 27, lines 17-29; page 30, line 24 – page 31, line 12; page 32, line 24 – page 33, line 19; FIG. 3. Finally, the instruction set includes a command to load a set of selected elements of the precharged row pointed to by said row address register into a selected set of said data registers (*e.g.*, registers 0-k in register file 112), said selection of elements based on bits in said bit mask (*e.g.*, 108), wherein all the

selected elements of the precharged row are loaded into the selected sets of data registers in a single operation. *See, e.g., id.* at page 8, line 18 – page 9, line 13; page 10, lines 17-21; page 28, line 21 – page 30, line 12; FIGS. 2-3.

With regard to the aspect of the invention set forth in independent claim 16, discussions of the recited features of claim 16 can be found at least in the below cited locations of the specification and drawings. By way of example, an embodiment in accordance with claim 16 relates to an embedded-DRAM processor. *See, e.g., id.* at page 4, lines 19-21; FIG. 1. The embedded-DRAM processor includes an embedded DRAM array (*e.g.*, 102, 534) having a plurality of random access memory cells arranged in rows and columns. *See, e.g., id.* at page 6, lines 22-26; page 7, lines 7-10; page 8, lines 7-9 and 18-20; page 9, lines 14-20 and 26-29; page 10, lines 15-18; page 13, lines 16-18; page 22, lines 7-23; page 36, lines 12-22; FIGS. 1, 4-5. The embedded-DRAM processor further includes a row address register (*e.g.*, 302) that holds a pointer that points to a row (*e.g.*, 208) of the DRAM array (*e.g.*, 102, 534). *See, e.g., id.* at page 7, lines 7-16; page 8, lines 18-28; page 26, lines 5-24; page 31, lines 13-17; page 33, lines 12-19; FIGS. 1, 3-4. The embedded-DRAM processor further includes first and second register files (*e.g.*, 112 and 114), each of said register files having a plurality of data registers (*e.g.*, registers 0-k in register file 112) capable of loading or storing an entire row (*e.g.*, 208) of the DRAM array (*e.g.*, 102, 534) in response to a single latch signal, each of the register files (*e.g.*, 112 and 114) also being capable of being placed into an active state and an inactive state (*e.g.*, inactive register file 408). *See, e.g., id.* at page 7, line 7 – page 8, line 22; page 10, lines 17-21; page 24, lines 18-29; page 27, lines 19-30; page 28, line 21 – page 30, line 12; FIGS. 1-4. The embedded-DRAM processor further includes a set of functional units (*e.g.*, 128) that perform logical operations on data accessed from a set of architectural registers, wherein registers placed into the active state appear as architectural registers to a set of functional units (*e.g.*, 128), and registers in the inactive state (*e.g.*, 408) are not accessible by the functional units (*e.g.* 128). *See, e.g., id.* at page 6, lines 6-21; page 8, line 28 – page 9, lines 13; FIGS. 1-2, 4. The embedded-DRAM processor further

includes a bit mask (*e.g.*, 108) to select one or more data locations within at least one of said register sets (*e.g.*, registers 0-k in register file 112). *See, e.g., id.* at page 6, line 30 – page 7, line 6; page 7, line 16 – page 8, line 6; page 8, line 18 – page 9, line 13; page 16, lines 5-11; page 22, line 8 – page 23, line 2; page 26, lines 19-23; page 26, line 24 – page 27, line 6; page 28, lines 4-20; page 29, lines 3-11; FIGS. 1-2. The embedded-DRAM processor additionally includes an instruction set having a command to perform arithmetic operations on said row address register (*e.g.*, 302). *See, e.g., id.* at page 8, line 18 – page 9, line 13; page 10, lines 22-28; page 11, lines 1-25; page 24, lines 18-21; FIG. 3. Finally, the instruction set includes a command to load a set of selected elements of the row pointed to by said row address register into a selected set of said data registers (*e.g.*, registers 0-k in register file 112), said selection of elements based on bits in said bit mask (*e.g.*, 108), wherein all the selected elements of the row (*e.g.*, 208) are loaded into the selected sets of data registers in a single operation, the selected set of said data registers being in the inactive state (*e.g.*, 408). *See, e.g., id.* at page 6, lines 6-21; page 8, line 18 – page 9, line 13; page 10, lines 17-21; page 28, line 21 – page 30, line 12; FIGS. 1-4.

With regard to the aspect of the invention set forth in independent claim 23, discussions of the recited features of claim 23 can be found at least in the below cited locations of the specification and drawings. By way of example, an embodiment in accordance with claim 23 relates to an embedded-DRAM processor. *See, e.g., id.* at page 4, lines 19-21; FIG. 1. The embedded-DRAM processor includes an embedded DRAM array (*e.g.*, 102, 534) having a plurality of random access memory cells arranged in rows and columns. *See, e.g., id.* at page 6, lines 22-26; page 7, lines 7-10; page 8, lines 7-9 and 18-20; page 9, lines 14-20 and 26-29; page 10, lines 15-18; page 13, lines 16-18; page 22, lines 7-23; page 36, lines 12-22; FIGS. 1, 4-5. The embedded-DRAM processor further includes a row address register (*e.g.*, 302) that holds a pointer that points to a row (*e.g.*, 208) of the DRAM array (*e.g.*, 102, 534). *See, e.g., id.* at page 7, lines 7-16; page 8, lines 18-28; page 26, lines 5-24; page 31, lines 13-17; page 33, lines 12-19; FIGS. 1, 3-4. The

embedded-DRAM processor further includes first and second registers files (*e.g.*, 112 and 114), each of said register files capable of loading or storing an entire row (*e.g.*, 208) of the DRAM array (*e.g.*, 102, 534) in response to a single latch signal. *See, e.g., id.* at page 7, line 7 – page 8, line 22; page 10, lines 17-21; page 24, lines 18-29; page 27, lines 19-30; page 28, line 21 – page 30, line 12; FIGS. 1-4. The embedded-DRAM processor further includes an embedded processor comprising first and second functional units (*e.g.*, 128), the first and second functional units having respective first and second instruction sets and capable of accessing said first and second register files (*e.g.*, 112 and 114). *See, e.g., id.* at page 8, line 18 – page 9, line 13; page 11, line 26 – page 12, line 9; page 12, line 27 – page 13, line 15; page 16, line 23 – page 17, line 18; FIGS. 1-2. Further, the first and second register files include a parallel access port (*e.g.*, 204) operative to parallely transfer contents of one of said register files between a DRAM row (*e.g.*, 208) as selected by said row-address register (*e.g.*, 302), the first and second register files (*e.g.*, 112 and 114) further include at least a second access port (*e.g.*, port on switch 204 connecting register file to 206) operative to transfer data between a selected one of said register files and the second functional unit (*e.g.*, 128). *See, e.g.,* at page 6, line 22 – page 7, line 6; page 7, lines 13-18; page 8, lines 7-17; page 10; lines 18-30; page 28, line 4 – page 30, line 8; FIG. 2. Additionally, the first instruction set includes a command to manipulate data in a data register (*e.g.*, registers 0-k) within a register file (*e.g.*, 112). *See, e.g., id.* at page 8, lines 7-17; page 11, lines 1-14; page 15, lines 8-28; page 28, line 21 – page 29, line 23; FIG. 2. Finally, second instruction set includes a command to perform an arithmetic operation on said row address register (*e.g.*, 302) and a command to load the entire row (*e.g.*, 208) pointed to by said row address register into a selected set of registers of said register files in a single operation. *See, e.g., id.* at page 6, lines 6-21; page 8, line 18 – page 9, line 13; page 10, lines 17-28; page 11, lines 1-25; page 24, lines 18-21; page 28, line 21 – page 30, line 12; FIGS. 2-3.

With regard to the aspect of the invention set forth in independent claim 28, discussions of the recited features of claim 28 can be found at least in the below cited

locations of the specification and drawings. By way of example, an embodiment in accordance with claim 28 relates to an embedded-DRAM processor. *See, e.g., id.* at page 4, lines 19-21; FIG. 1. The embedded-DRAM processor includes an embedded DRAM array (*e.g.*, 102, 534) having a plurality of random access memory cells arranged in rows and columns. *See, e.g., id.* at page 6, lines 22-26; page 7, lines 7-10; page 8, lines 7-9 and 18-20; page 9, lines 14-20 and 26-29; page 10, lines 15-18; page 13, lines 16-18; page 22, lines 7-23; page 36, lines 12-22; FIGS. 1, 4-5. The embedded-DRAM processor further includes first and second dual-port (*e.g.*, provided by switches 204) registers files (*e.g.*, 112, 114), each of said register files capable of parallely transferring data between an entire row (*e.g.*, 208) of said DRAM array (*e.g.*, 102, 534) in a single operation, each of the register files (*e.g.*, 112, 114) also being capable of being placed into an active state and an inactive state (*e.g.*, inactive register file 408). *See, e.g., id.* at page 7, line 7 – page 8, line 22; page 10, lines 17-21; page 24, lines 18-29; page 27, lines 19-30; page 28, line 21 – page 30, line 12; FIGS. 1-4. The embedded-DRAM processor further includes first and second embedded functional units (*e.g.*, 128), the first and second functional units having respective first and second instruction sets that operate on registers of an architectural register set, the architectural register set including one of the first and second dual-port register files (*e.g.*, 112 and 114) that is currently in the active state (*e.g.*, 408). *See, e.g., id.* at page 6, lines 6-21; page 8, line 18 – page 9, line 13; page 11, lines 1-14; page 11, line 26 – page 12, line 9; page 12, line 27 – page 13, line 15; page 16, line 23 – page 17, line 18; FIGS. 1-2, 4. Additionally, the first instruction set includes a command to manipulate data in a data register (*e.g.*, registers 0-k) within a register file (*e.g.*, 112). *See, e.g., id.* at page 8, lines 7-17; page 11, lines 1-14; page 15, lines 8-28; page 28, line 21 – page 29, line 23; FIG. 2. Finally, the second instruction set includes a command to unidirectionally transfer data between an entire row (*e.g.*, 208) of said DRAM array (*e.g.*, 102, 534) and a selected inactive data register file (*e.g.*, 408), wherein the transfer occurs in a single operation and a command to place the selected inactive data register file (*e.g.*, 408) into said active state, wherein when the inactive register file is activated, it becomes an architectural register set of said first functional unit. *See, e.g., id.* at page 6, lines 6-21;

page 8, line 18 – page 9, line 13; page 10, lines 17-28; page 11, lines 1-25; page 24, line 18 – page 25, line 20; page 28, line 21 – page 30, line 12; page 40, lines 23-25; FIGS. 2-4.

With regard to the aspect of the invention set forth in independent claim 45, discussions of the recited features of claim 45 can be found at least in the below cited locations of the specification and drawings. By way of example, an embodiment in accordance with claim 45 relates to an embedded-DRAM processor. *See, e.g., id.* at page 4, lines 19-21; FIG. 1. The embedded-DRAM processor includes an embedded DRAM array (*e.g.*, 102, 534) having a plurality of random access memory cells. *See, e.g., id.* at page 6, lines 22-26; page 7, lines 7-10; page 8, lines 7-9 and 18-20; page 9, lines 14-20 and 26-29; page 10, lines 15-18; page 13, lines 16-18; page 22, lines 7-23; page 36, lines 12-22; FIGS. 1, 4-5. The embedded-DRAM processor further includes first and second dual-port (*e.g.*, provided by switches 204) registers files (*e.g.*, 112, 114), whereby the first port of each of the register files is a parallel access port and is parallely coupled to the DRAM array (*e.g.*, 102, 534), and wherein each of the register files are capable of being placed into an active state and an inactive state (*e.g.*, inactive register file 408). *See, e.g., id.* at page 7, line 7 – page 8, line 22; page 10, lines 17-21; page 24, lines 18-29; page 27, lines 19-30; page 28, line 21 – page 30, line 12; FIGS. 1-4. Additionally, the embedded-DRAM processor includes a functional unit (*e.g.*, 128) that executes a first program, said functional unit coupled (*e.g.*, by way of element 206) to said second port (*e.g.*, second port of 204) of said register files (*e.g.*, 112, 114), wherein the functional unit (*e.g.*, 128) is responsive to commands exclusively involving architectural register operands that map onto the registers within a register file that is in the active state. *See, e.g., id.* at page 11, lines 26 – page 12, line 9; page 13, line 16 – page 14, line 21; page 24, line 18 – page 25, line 20; page 30, lines 7-23; FIGS. 1-2. Finally, the embedded-DRAM processor includes a data assembly unit (*e.g.*, 122) responsive to an instruction set. *See, e.g., id.* at page 7, line 7 – page 8, line 6; page 11, lines 1-14; page 12, line 27 – page 13, line 15; page 23, line 3 – page 24, line 5; FIGS. 1-2. The instruction set includes a command that causes

data to be moved between the DRAM array (*e.g.*, 102, 534) and a register file that is in the inactive state (*e.g.*, 408) and a command that causes said register file in the inactive state to assume the active state and said register file in the active state to assume the inactive state. *See, e.g., id.* at page 6, lines 6-21; page 24, line 18 – page 25, line 20; page 29, line 21 – page 30, line 23; FIGS. 1, 4.

With regard to the aspect of the invention set forth in independent claim 46, discussions of the recited features of claim 46 can be found at least in the below cited locations of the specification and drawings. By way of example, an embodiment in accordance with claim 46 relates to a method for use in a digital processor having an embedded DRAM array (*e.g.*, 102, 534) having a plurality of random access memory cells arranged in rows and columns and at least one row address register (*e.g.*, 302), wherein the at least one row address register is capable of being loaded or stored. *See, e.g., id.* at page 4, lines 19-21; page 6, lines 22-26; page 7, lines 7-10; page 8, lines 7-9 and 18-20; page 9, lines 14-20 and 26-29; page 10, lines 15-18; page 13, lines 16-18; page 22, lines 7-23; page 36, lines 12-22; FIGS. 1, 4-5. The method includes performing an arithmetic operation (*e.g.*, using row address arithmetic unit 306) on the at least one row address register (*e.g.*, 302) in order to manipulate a pointer that points to a row of the embedded DRAM array. *See, e.g., id.* at page 10, line 15 – page 11, line 25; page 31, line 13 – page 32, line 7; FIG. 3. The method further includes speculatively precharging (activating) a row pointed to by said at least one row address register (*e.g.*, 302) based at least partially upon historical program execution data indicating a possible need to perform one or more load or store operations that would access said row. *See, e.g., id.* at page 14, line 10 – page 15, line 7; page 24, line 18 – page 26, line 23; page 30, lines 7-23; page 31, line 13 – page 32, line 23; FIG. 3. Additionally, the method includes, in response to a separate command executed after the speculatively precharging, loading a plurality of words of a row designated by the at least one row address register (*e.g.*, 302) into designated sets of data registers (*e.g.*, registers 0-k) in a single operation. *See, e.g.,*

id. at page 6, lines 6-21; page 8, line 18 – page 9, line 13; page 10, lines 17-28; page 11, lines 1-25; page 24, lines 18-21; page 28, line 21 – page 30, line 12; FIGS. 2-3.

With regard to the aspect of the invention set forth in independent claim 49, discussions of the recited features of claim 49 can be found at least in the below cited locations of the specification and drawings. By way of example, an embodiment in accordance with claim 49 relates to a method for use in a digital processor having an embedded DRAM array (*e.g.*, 102, 534) that includes a plurality of random access memory cells arranged in rows and columns, first and second dual-port (*e.g.*, provided by switches 204) registers files (*e.g.*, 112, 114), wherein each of the register files are capable of parallely transferring of data between an entire row (*e.g.*, 208) of the embedded DRAM array (*e.g.*, 102, 534) in a single operation, wherein each of the register files are also capable of being placed into an active state and an inactive state (*e.g.*, 408), and wherein the digital processor further includes first and second functional units (*e.g.*, 128). *See, e.g., id.* at page 4, lines 19-21; page 6, line 22 – page 9, line 29; page 10, lines 15-30; page 11, line 26 – page 12, line 9; page 12, line 27 – page 13, line 18; page 16, line 23 – page 17, line 18; page 22, lines 7-23; page 24, lines 18-29; page 27, lines 19-30; page 28, line 4 – page 30, line 12; page 36, lines 12-22; FIGS. 1-5. The method includes manipulating data in a data register (*e.g.*, registers 0-k) within a register file (*e.g.*, 112) that is in the active state using said first functional unit. *See, e.g., id.* at page 8, lines 7-17; page 11, lines 1-14; page 15, lines 8-28; page 28, line 21 – page 29, line 23; FIG. 2. The method further includes using the second functional unit to unidirectionally transferring data between an entire row (*e.g.*, 208) of said embedded DRAM array (*e.g.*, 102, 534) and a selected inactive data register file (*e.g.*, 408), wherein the transfer of the data occurs in a single operation, and placing the inactive register file (*e.g.*, 408) into the active state, whereby when the register file is activated, it becomes an architectural register set of the first functional unit. *See, e.g., id.* at page 6, lines 6-21; page 8, line 18 – page 9, line 13; page 10, lines 17-28; page 11, lines 1-25; page 24, line 18 – page 25, line 20; page 28, line 21 – page 30, line 12; page 40, lines 23-25; FIGS. 2-4.

With regard to the aspect of the invention set forth in independent claim 50, discussions of the recited features of claim 50 can be found at least in the below cited locations of the specification and drawings. By way of example, an embodiment in accordance with claim 50 relates to an embedded-DRAM processor. *See, e.g., id.* at page 4, lines 19-21; FIG. 1. The embedded-DRAM processor includes an embedded DRAM array (*e.g.*, 102, 534) having a plurality of random access memory cells arranged in rows and columns. *See, e.g., id.* at page 6, lines 22-26; page 7, lines 7-10; page 8, lines 7-9 and 18-20; page 9, lines 14-20 and 26-29; page 10, lines 15-18; page 13, lines 16-18; page 22, lines 7-23; page 36, lines 12-22; FIGS. 1, 4-5. The embedded-DRAM processor further includes an embedded row address register (*e.g.*, 302) that holds a pointer that points to a row (*e.g.*, 208) of the DRAM array (*e.g.*, 102, 534). *See, e.g., id.* at page 7, lines 7-16; page 8, lines 18-28; page 26, lines 5-24; page 31, lines 13-17; page 33, lines 12-19; FIGS. 1, 3-4. The embedded-DRAM processor further includes embedded first and second register files (*e.g.*, 112 and 114) capable of loading or storing an entire row (*e.g.*, 208) of the DRAM array (*e.g.*, 102, 534) in response to a single latch signal, each of the register files (*e.g.*, 112 and 114) also being capable of being placed into an active state and an inactive state (*e.g.*, inactive register file 408). *See, e.g., id.* at page 7, line 7 – page 8, line 22; page 10, lines 17-21; page 24, lines 18-29; page 27, lines 19-30; page 28, line 21 – page 30, line 12; FIGS. 1-4. The embedded-DRAM processor further includes embedded first and second functional units (*e.g.*, 128), the first and second functional units having respective first and second instruction sets and being capable of accessing said first and second register files (*e.g.*, 112 and 114) while in the first and second registers are in the active state. *See, e.g., id.* at page 8, line 18 – page 9, line 13; page 11, line 26 – page 12, line 9; page 12, line 27 – page 13, line 15; page 16, line 23 – page 17, line 18; FIGS. 1-2. Further, the first and second register files include a parallel access port (*e.g.*, 204) operative to parallely transfer contents of one of said register files between a DRAM row (*e.g.*, 208) as selected by said row-address register (*e.g.*, 302), wherein each of the register files (*e.g.*, 112 and 114) further includes at least a second access port (*e.g.*, port on switch

204 connecting register file to 206) operative to transfer data between a selected register files and the second functional unit (*e.g.*, 128). *See, e.g.*, at page 6, line 22 – page 7, line 6; page 7, lines 13-18; page 8, lines 7-17; page 10; lines 18-30; page 28, line 4 – page 30, line 8; FIG. 2. Additionally, the first instruction set includes a command to manipulate data in a data register (*e.g.*, registers 0-k) within a register file (*e.g.*, 112). *See, e.g., id.* at page 8, lines 7-17; page 11, lines 1-14; page 15, lines 8-28; page 28, line 21 – page 29, line 23; FIG. 2. Finally, second instruction set includes a command to perform an arithmetic operation on said row address register (*e.g.*, 302) and a command to load the entire row (*e.g.*, 208) pointed to by said row address register (*e.g.*, 302) into a selected set of registers of the register files that are currently in the inactive state in a single operation. *See, e.g., id.* at page 6, lines 6-21; page 8, line 18 – page 9, line 13; page 10, lines 17-28; page 11, lines 1-25; page 24, lines 18-21; page 28, line 21 – page 30, line 12; FIGS. 2-3.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Due the number of independent claims and the nature of the rejections, Appellant has provided the Board with the following brief summary regarding the subject matter argued below.

- First, the cited references, taken alone or in hypothetical combination, do not disclose or suggest register files capable of loading or storing an entire row of a DRAM array in a single operation or in response to a single latch signal, nor do the cited references teach or suggest a command for doing the same, as generally recited by independent claims 1, 13, 16, 23, 28, 46, 49, and 50. Appellant's arguments addressing this deficiency are presented in Sections VII.A.2, VII.B.1, VII.E.1, and VII.G.1 of the Appeal Brief.

- Second, the cited references, taken alone or in hypothetical combination, do not disclose or suggest dual-port register files having a first port operative to transfer data between a register file and a row of a DRAM array and a second port operative to transfer data between the register file and a functional unit, as recited by independent claims 23 and 50. Appellant's arguments addressing

this deficiency are presented in Sections VII.B.2 and VII.E.2 of the Appeal Brief.

- Third, the cited references, taken alone or in hypothetical combination, do not disclose or suggest a dual-port register file having a first access port coupled to a DRAM array and configured to parallely transfer data between the register file and the DRAM array, as recited by independent claim 45. Appellant's arguments addressing this deficiency are presented in Section VII.E.3 of the Appeal Brief.

- Fourth, the cited references, taken alone or in hypothetical combination, do not disclose or suggest a command for deactivating a row of a DRAM array after it had been previously precharged, as recited by independent claim 1. Appellant's arguments addressing this deficiency are presented below in Section VII.A.3 of the Appeal Brief.

- Fifth, the cited references, taken alone or in hypothetical combination, do not disclose or suggest register files capable of being placed in an inactive and active state, nor do the cited references teach or suggest a command for causing register files to transition between the inactive and active states, as recited by independent claims 16, 28, 45, and 49. Appellant's arguments addressing this deficiency are presented below in Section VII.E.4 of the Appeal Brief.

A. First Ground of Rejection for Review on Appeal

Appellant respectfully urges the Board to review and reverse the Examiner's first ground of rejection in which the Examiner rejected claims 1, 6-10, 12-14, and 51 under 35 U.S.C. § 103(a) as being unpatentable over the combination of Inagami et al., U.S. Patent No. 4,881,168 (hereinafter the "Inagami reference"), Luk et al., U.S. Patent No. 5,883,814 (hereinafter the "Luk reference"), and Farmwald et al., U.S. Patent No. 5,243,703 (hereinafter the "Farmwald reference"), with Garcia, U.S. Patent No. 4,827,476 (hereinafter the "Garcia reference") cited as extrinsic evidence.

B. Second Ground of Rejection for Review on Appeal

Appellant respectfully urges the Board to review and reverse the Examiner's second ground of rejection in which the Examiner rejected claims 23-25 under 35 U.S.C. § 103(a) as being unpatentable over the combination of the Inagami and Luk references.

C. Third Ground of Rejection for Review on Appeal

Appellant respectfully urges the Board to review and reverse the Examiner's third ground of rejection in which the Examiner rejected claims 2-5 and 11 under 35 U.S.C. § 103(a) as being unpatentable over the combination of the Inagami, Luk and Farmwald references, and further in view of Parady, U.S. Patent No. 5,933,627 (hereinafter the "Parady reference").

D. Fourth Ground of Rejection for Review on Appeal

Appellant respectfully urges the Board to review and reverse the Examiner's fourth ground of rejection in which the Examiner rejected claims 26 and 27 under 35 U.S.C. § 103(a) as being unpatentable over the combination of the Inagami, Luk, and Parady references.

E. Fifth Ground of Rejection for Review on Appeal

Appellant respectfully urges the Board to review and reverse the Examiner's fifth ground of rejection in which the Examiner rejected claims 16-22, 28-32, 34-43, 45, and 49-50 under 35 U.S.C. § 103(a) as being unpatentable over the combination of the Inagami, Luk, and Parady references, and further in view of Bissett et al., U.S. Patent No. 5,896,523 (hereinafter the "Bissett reference").

F. Sixth Ground of Rejection for Review on Appeal

Appellant respectfully urges the Board to review and reverse the Examiner's sixth ground of rejection in which the Examiner rejected claim 44 under 35 U.S.C. § 103(a) as

being unpatentable over the combination of the Inagami, Luk, Parady, and Bissett references, and further in view of the Farmwald reference.

G. Seventh Ground of Rejection for Review on Appeal

Appellant respectfully urges the Board to review and reverse the Examiner's seventh ground of rejection in which the Examiner rejected claim 46 under 35 U.S.C. § 103(a) as being unpatentable over the combination of the Inagami and Luk references, and further in view of Krishnamohan et al., U.S. Patent No. 5,499,355 (hereinafter the "Krishnamohan reference").

H. Eighth Ground of Rejection for Review on Appeal

Appellant respectfully urges the Board to review and reverse the Examiner's eighth ground of rejection in which the Examiner rejected claim 47 under 35 U.S.C. § 103(a) as being unpatentable over the combination of the Inagami, Luk, and Krishnamohan references, and further in view of the Farmwald reference, with the Garcia reference cited as extrinsic evidence.

VII. ARGUMENT

As will be discussed in detail below, the Examiner has improperly rejected the pending claims. Moreover, the Examiner has misapplied long-standing and binding legal precedents and principles in rejecting the claims under Section 103. Accordingly, Appellant respectfully requests full and favorable consideration by the Board, as Appellant strongly believes that independent claims 1, 13, 16, 23, 28, 45, 46, 49, and 50, as well as their respective dependent claims, are presently in condition for allowance.

A. First Ground of Rejection

The Examiner rejected claims 1, 6-10, 12-14, and 51 under 35 U.S.C. § 103(a) as being unpatentable based on the combination of the Inagami, Luk, and Farmwald references, with the Garcia reference further cited as extrinsic evidence with regard to

DRAM refresh cycles. Of these, claims 1 and 13 are independent. Appellant respectfully traverses this rejection.

1. Judicial precedent has clearly established a legal standard for an obviousness rejection under Section 103.

Before continuing, Appellant respectfully notes that all the grounds of rejection set forth by the Examiner in the Final Office Action and presented for review in the present Appeal are based upon Section 103. Accordingly, it should be understood that the legal guidelines provided in the present discussion regarding the Examiner's rejection of claims 1, 6-10, 12-14, and 51 are also applicable to the remaining grounds of rejections discussed below. Therefore, in the interest of brevity and to avoid redundancy, Appellant has not reproduced the following text for the discussion of each of the subsequent grounds of rejection herein, but rather respectfully requests that the Board keep in mind these legal guidelines regarding obviousness rejections under Section 103 when considering Appellant's arguments with regard to each of the remaining grounds of rejection currently being appealed.

With this in mind, Appellant notes that it is well established case law that the burden of establishing a *prima facie* case of obviousness falls on the Examiner. *Ex parte Wolters and Kuypers*, 214 U.S.P.Q. 735 (PTO Bd. App. 1979). To establish a *prima facie* case, the Examiner must show, among other things, that the combination includes *all* of the claimed elements. *Ex parte Clapp*, 227 U.S.P.Q. 972 (B.P.A.I. 1985). (Emphasis added.) Furthermore, in addressing obviousness determinations under 35 U.S.C. § 103, the Supreme Court in *KSR International Co. v. Teleflex Inc.*, 127 S. Ct. 1721 (2007), reaffirmed many of its precedents relating to obviousness including its holding in *Graham v. John Deere Co.*, 383 U.S. 1 (1966). In *KSR*, the Court also reaffirmed that "a patent composed of several elements is not proved obvious merely by demonstrating that each of its elements was, independently, known in the prior art." *Id.* at 1741. In this regard, the *KSR* court stated that "it can be important to identify a reason

that would have prompted a person of ordinary skill in the relevant field to combine the elements in the way the claimed new invention does … because inventions in most, if not all, instances rely upon building blocks long since uncovered, and claimed discoveries almost of necessity will be combinations of what, in some sense, is already known." *Id.* In *KSR*, the court noted that the demonstration of a teaching, suggestion, or motivation to combine provides a "helpful insight" in determining whether claimed subject matter is obvious. *Id.*

Furthermore, the *KSR* court did not diminish the requirement for objective evidence of obviousness. *Id.* ("To facilitate review, this analysis should be made explicit. *See In re Kahn,* 441 F.3d 977, 988 (CA Fed. 2006) ('[R]ejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness')"). As our precedents make clear, however, the analysis need not seek out precise teachings directed to the specific subject matter of the challenged claim, for a court can take account of the inferences and creative steps that a person of ordinary skill in the art would employ."); *see also In re Lee,* 61 U.S.P.Q.2d 1430, 1436 (Fed. Cir. 2002) (holding that the factual inquiry whether to combine references must be thorough and searching, and that it must be based on objective evidence of record).

Moreover, it is well established case law that pending claims must be given an interpretation that is reasonable and consistent with the *specification*. *See In re Prater*, 415 F.2d 1393, 1404-05, 162 U.S.P.Q. 541, 550-51 (C.C.P.A. 1969) (emphasis added); *see also In re Morris,* 127 F.3d 1048, 1054-55, 44 U.S.P.Q.2d 1023, 1027-28 (Fed. Cir. 1997); *see also* M.P.E.P. §§ 608.01(o) and 2111. Indeed, the specification is "the primary basis for construing the claims." *See Phillips v. AWH Corp.*, No. 03-1269, -1286, at 13-16 (Fed. Cir. July 12, 2005) (*en banc*). One should rely *heavily* on the written description for guidance as to the meaning of the claims. *See id.* Moreover, interpretation of the claims must *also* be consistent with the interpretation that *one of ordinary skill in the art*

would reach. *See In re Cortright*, 165 F.3d 1353, 1359, 49 U.S.P.Q.2d 1464, 1468 (Fed. Cir. 1999); M.P.E.P. § 2111. "The inquiry into how a person of ordinary skill in the art understands a claim term provides an objective baseline from which to begin claim interpretation." *See Collegenet, Inc. v. ApplyYourself, Inc.*, 418 F.3d 1225, 75 U.S.P.Q.2d 1733, 1738 (Fed. Cir. 2005) (quoting *Phillips v. AWH Corp.*, 75 U.S.P.Q.2d 1321, 1326). The Federal Circuit has made clear that derivation of a claim term must be based on "usage in the ordinary and accustomed meaning of the words amongst artisans of ordinary skill in the relevant art." *See id.* Accordingly, if the Examiner's interpretation of a recited claim term is neither consistent with the specification, nor consistent with the interpretation that one skilled in the art would reach, the interpretation cannot be considered reasonable.

With these legal tenets in mind, Appellant will now address the numerous deficiencies of the Examiner's rejections below.

2. The cited references, alone or in combination, fail to teach or suggest sets of data registers (register files) capable of loading or storing either an entire row of a DRAM array or selected columns of an entire row of a DRAM array either in a single operation or in response to a single latch signal, and further fail to teach or suggest a command for doing the same, as recited by independent claims 1 and 13.

As an initial note, Appellant notes that the Examiner cited the Inagami reference in the Office Action mailed August 21, 2007 as allegedly disclosing the loading/storing of a DRAM row by a register file in response to a single latch signal, a feature generally recited by each of independent claims 1, 13, 16, 23, 28, 46, 49, and 50. *See, generally*, Office Action mailed 8/21/2007. Although Appellant did not necessarily agree with these rejections, Appellant chose to amend independent claims 1, 13, 16, 23, 28, 46, 49, and 50, as indicated in the Response filed November 21, 2007 (hereinafter the "Previous Response"), for clarification of certain features in order to more clearly distinguish these

claims from the Inagami reference. *See* Previous Response, pages 3-18. Specifically, each of these claims was amended to clarify that the recited register file or sets of data registers are capable of loading/storing an entire row (or all the selected columns of an entire row) of a DRAM array in a single operation or in response to a single latch signal.

Accordingly, independent claim 1 recites, *inter alia*, "one or more sets of registers, each of said sets of registers capable of loading or storing an *entire row* of the DRAM array in response to a single latch signal" and "a command to load selected columns of the precharged row into designated sets of data registers … wherein all the selected columns of the precharged row are loaded into the designated sets of data registers in a single operation." (Emphasis added.) Independent claim 13 recites, *inter alia*, "one or more sets of data registers, each of said sets of data registers capable of loading or storing an entire row of the DRAM array in response to a single latch signal" and "a command to load a set of selected elements of the precharged row pointed to by said row address register into a selected set of said data registers … wherein all the selected elements of the precharged row are loaded into the selected sets of data registers in a single operation." (Emphasis added.)

Appellant respectfully submits that the Inagami reference (which appears to be the sole reference relied upon by the Examiner in the Final Office Action as allegedly teaching the above features) fails to disclose at least these features of independent claims 1 and 13. In stark contrast to the recited subject matter, the Inagami reference discloses a DRAM system in which at least four load/store operations (and thus four latch signals, one latch signal for each load operation) are required to load or store an entire row of a DRAM into a register file. In setting forth the present rejection, the Examiner, referring to FIG. 4 of the Inagami reference, asserted the row 21 as being the recited "entire row" of a DRAM array, the vector register file 23 as being the recited "sets of data registers," (*e.g.*, register files), and the main storage 1 as being the recited "DRAM array." *See* Final Office Action, pages 3-5, 9-10. However, as discussed in detail by Appellant in the

Previous Response, even assuming such an interpretation is reasonable, FIGS. 4 and 5 of the Inagami reference clearly illustrate the vector register file 23 as being divided into four storage locations, each storage location having four elements, which the Examiner has correlated to data registers within the vector register file 23. *See* Previous Response, pages 30-34; *see also* Inagami, FIGS. 4-5. The Inagami reference further discloses that the four storage locations within the vector register file 23 may be configured to load data from a row 21 of the main storage 1 using multiple separate load operations. *See id.* For example, depending on the vector mask 22, the first storage location may be configured to load/store elements a0-a3, the second storage location may be configured to load/store elements a4-a7, and so forth. Thus, as best understood from the teachings of Inagami, the loading or storing of an entire row (*e.g.*, row 21) of the main storage 1 into the register file 23 requires that the first four elements are loaded into the first storage location, the second four elements are loaded into the second storage location, and so forth.

Keeping these points in mind, the Inagami reference also makes it clear that the operations for loading/storing of each of these storage locations are *separate and distinct* from each other. For example, Inagami discloses that in a first load operation, vector elements a0 to a3 are compared with a vector mask 22 and are loaded accordingly by the load/store sub-pipes 2-0 to 2-3. *See id.* at col. 6, lines 48-53. Further, in a second load operation, vector elements a4 to a7 are compared with the vector mask 22 and loaded accordingly in a similar manner as the first load operation. *See id.* at col. 6, lines 53-63 (note, however, that only a4-a6 are loaded due to the vector mask). A third load operation follows, loading the next four elements, along with a fourth load operation for loading the last four elements. *See id.* at col. 6, line 63 – col. 7, line 8. Appellant further notes that FIG. 5 of the Inagami reference shows a similar technique for executing a series of store operations (essentially the reverse of the load operations described in FIG. 4), wherein data from the vector register 32 is stored into a row 21 of the main storage 1 (designated by vector data structure 12) by four separate and distinct store operations. *See id.* at col. 7, lines 14-36; FIG. 5. Indeed, the Inagami reference is *explicit* that at least

four separate operations are required to load or store an entire row (*e.g.*, row 21) of a DRAM array into the vector register file 23, which the Examiner asserted as the recited "sets of registers" (*e.g.*, register file). Thus, in stark contrast to the subject matter recited by independent claims 1 and 13, which requires a set of registers (*e.g.*, register file) being capable of loading or storing an entire row of a DRAM array in a single operation (or a command for doing the same), it is abundantly clear that the vector registers 23 (*e.g.*, asserted register file) described by the Inagami reference requires a plurality of separate and distinct operations for loading and storing an entire row of a DRAM array.

Moreover, because the Inagami reference *clearly* requires at least four separate load or store operations, Appellant submits that those skilled in the art will readily appreciate that a separate latch signal is associated with each of the four operations to initiate the transfer of each group (i.e., elements a0-a3) of data. Therefore, it is believed that the Inagami reference also fails to teach that an entire row of the DRAM is loaded or stored by a register file in response to a single latch signal. Moreover, even if it can be argued that the multiple load operations taught by the Inagami reference occur in parallel (*e.g.*, at the same time), loading the entire row of data into the vector register files 23 still requires four separate load operations. As such, Appellant does not believe that the Inagami reference could reasonably be construed as disclosing the loading and storing of an entire row of a DRAM array in a single operation or in response to a single latch signal, as set forth by independent claims 1 and 13.

In the Final Office Action, the Examiner acknowledged that the DRAM system disclosed by the Inagami reference does require the plurality of load/store operations to transfer an entire row of data, as discussed above, but asserted that Appellant's interpretation of the term "entire row" is too broad. *See* Final Office Action, page 53. The Examiner further stated that "[Appellant], when applying such an interpretation, **does make a valid argument**." *Id.* (Emphasis added.) However, the Examiner further

insisted that the phrase "entire row" should be interpreted to mean a subset of the row 21, or even an individual element or cell. *See id.* Specifically, the Examiner asserted:

> ...a single DRAM array row may correspond to a single entry shown in storage 21 in Fig. 4. For instance, data a0 is in row 0, data a1 is in row 1, etc. Or, in an alternate interpretation, a single DRAM row may correspond to four consecutive entries shown in storage 21 in Fig. 4. *A row is merely a number of items arranged in a line. Hence, any number of data items in storage 21 can be called a row.*
>
> *Id.* (Emphasis added.)

As the Board will appreciate, the Federal Circuit has made it clear that the pending claims must not only be given an interpretation that is reasonable and consistent with the *specification*, but also be consistent with the interpretation that *one of ordinary skill in the art* would reach. *See In re Prater*, 415 F.2d 1393, 1404-05, 162 U.S.P.Q. 541, 550-51 (C.C.P.A. 1969) (emphasis added); *see also In re Morris,* 127 F.3d 1048, 1054-55, 44 U.S.P.Q.2d 1023, 1027-28 (Fed. Cir. 1997); *see also In re Cortright*, 165 F.3d 1353, 1359, 49 U.S.P.Q.2d 1464, 1468 (Fed. Cir. 1999); M.P.E.P. § 2111.

In view of the foregoing legal principles, Appellant strongly disagrees with the interpretation of the term "entire row" offered by the Examiner and respectfully submits that this proposed interpretation is blatantly inconsistent with the meaning that one skilled in the art would understand the term "row" to mean when applied to a memory device. Indeed, as will be appreciated, the term "row" has a very *specific* meaning when used in the context of memory devices. For instance, memory devices are typically constructed as an array of storage elements, commonly referred to as "memory cells," arranged in both a horizontal and vertical manner, wherein the horizontally arranged storage elements are referred to as "rows" and the vertically arranged storage elements are referred to as "columns." Further, when referring to a group of horizontally arranged storage elements as a "row," those of ordinary skill in the art will readily appreciate that the term "row" is regarded as meaning all or the entirety of the storage elements within a particular

horizontal arrangement of cells in a memory array. Thus, in the context of memory arrays, rows and columns have a very well-understood meaning and connotation. By reciting "entire row," it would be abundantly clear to those skilled in the art, the subject matter unambiguously claimed.

Further, as noted above, in response to the Examiner's rejection of this recited feature in the Office Action mailed August 21, 2007, Appellant *specifically* amended independent claims 1 and 13 to *clearly* specify that the recited "sets of data registers" (*e.g.*, register files) are capable of loading or storing an entire row in a single operation or in response to a single latch signal. Thus, even if the interpretation of the term "row" offered by the Examiner could be somehow be characterized as being reasonable in any way, one skilled in the art certainly would not interpret the addition of the term "**entire**" to claims 1 and 13 as suggesting a portion (*e.g.*, a single memory cell within a row, or a subset of an entire row) or anything *less* than an entire row.

With these points in mind, Appellant respectfully submits that the Inagami reference fails to teach or suggest the loading or storing of an entire row by a register file in a single operation or in response to a single latch signal, as recited by independent claims 1 and 13. Indeed, as discussed above, the Examiner's only rationale for maintaining the present rejection in the Final Office Action is based on the mistaken assumption that an "entire row" could somehow be reasonably interpreted as meaning a single memory cell within a row, or a subset of memory cells in a row. However, in view of the above discussion, this interpretation is clearly erroneous and unreasonable. Thus, contrary to the Examiner's assertions, Appellant respectfully submits to the Board that the Inagami reference does not teach or suggest at least this recited feature of independent claims 1 and 13. Moreover, the Examiner did not indicate that the Luk and Farmwald references, which were cited in combination with the Inagami reference, were relied upon in this regard. The Examiner also did not indicate that the Garcia reference, which was cited as extrinsic evidence with regard to DRAM refresh cycles, was relied upon to cure

the foregoing deficiencies. Thus, Appellant further asserts that each of these additionally cited references fails to obviate the above-discussed deficiencies of the Inagami reference. For at least this reason, Appellant respectfully submits that independent claims 1 and 13 are allowable over the combination of the Inagami, Luk, and Farmwald references, with the Garcia reference cited as extrinsic evidence.

3. The cited references, alone or in combination, do not teach or suggest a command to deactivate a row pointed to by a row address register after the row has been precharged, as recited by independent claim 1.

Independent claim 1 further recites "an instruction set which includes ... a command to precharge (activate) a row pointed to by [a] row address register" and "a command to deactivate a row pointed to by said row address register after it had been precharged by the row address register." (Emphasis added.) This recited feature is also believed to be absent missing from the cited combination of the Inagami, Luk, and Farmwald references, as well as from the Garcia reference which was cited as extrinsic evidence.

In the previous rejection of independent claim 1, the Examiner admitted that the Inagami reference fails to disclose this recited feature, but stated that the Farmwald reference allegedly cures this deficiency. *See* Office Action mailed 8/21/2007, page 5. Specifically, the Examiner stated:

> Inagami ... has not explicitly taught a command to deactivate said row pointed to by said row address register after it had been precharged by the command to precharge. However, it is known that any load that occurs in a DRAM is a destructive load. That is, reading a value in DRAM causes the charges making up that value to diminish. Consequently, after a value is read, the DRAM row must be refreshed. Farmwald has also taught a command which performs an explicit refresh. See Fig. 4 and column 12, lines 58-60.

Id. (Emphasis added.)

Based on the statements above, it appears that the Examiner is asserting a DRAM refresh command as being analogous to the "deactivate command" recited by independent claim 1. Appellant respectfully traverses the Examiner's rejection. In particular, Appellant submits that the Examiner has misapplied the long standing legal precedent, discussed above, which requires that the interpretation of pending claims must not only be reasonable and consistent with the specification, but also consistent with the interpretation that one of ordinary skill in the art would reach. *See In re Prater*, 415 F.2d 1393, 1404-05, 162 U.S.P.Q. 541, 550-51 (C.C.P.A. 1969) (emphasis added); *see also In re Cortright*, 165 F.3d 1353, 1359, 49 U.S.P.Q.2d 1464, 1468 (Fed. Cir. 1999); M.P.E.P. § 2111. Moreover, Appellant respectfully notes that the Federal Circuit, sitting *en banc*, recently provided a summary and additional guidance regarding the proper interpretation of claims in view of the specification. *See Phillips v. AWH Corp.*, 75 U.S.P.Q.2d 1321 (Fed. Cir. 2005) (*en banc*). In *Phillips*, the Federal Circuit again emphasized the primacy of the specification in claim interpretation. Particularly, the *Phillips* court noted that the specification "is always highly relevant to the claim construction analysis. Usually, it is dispositive; it is the single best guide to the meaning of a disputed term." *Phillips*, 75 U.S.P.Q.2d at 1327 (quoting *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1582 (Fed. Cir. 1996)) (Emphasis added.) Moreover, the court also noted that:

> Ultimately, the interpretation to be given a term can only be determined and confirmed with a full understanding of what the inventors actually invented and intended to envelop with the claim. The construction that stays true to the claim language *and most naturally aligns with the patent's description of the invention* will be, in the end, the correct construction.
>
> *Phillips*, 75 U.S.P.Q.2d at 1328-29. (Emphasis added.)

With the foregoing and controlling case law in mind, Appellant notes that the present Application *clearly* discloses that an embedded DRAM processor, in accordance

with an embodiment of the present invention, has an instruction set including *both* a command to *activate* (precharge) a row and a command to *deactivate* a row in a DRAM array. *See* Application, page 8, lines 22-27; page 11, lines 22-28; page 21, lines 3-5. The commands are utilized by a data assembly unit 122 which is operatively configured to control the movement of data between register files. *See id.* at page 21, lines 3-5. Furthermore, the activate and deactivate commands are based on a set of activation bits 204 which may be set or cleared under program control to activate or deactivate entire rows of *selected* DRAM banks in the DRAM arrays 102. *See id.* at page 26, lines 5-10; page 31, lines 13-20; FIG. 3. The data assembly unit 122 then assembles data corresponding to possible program branches into inactive register files so that the data required for each branch will be ready regardless of which branch a program ultimately takes. *See id.* at page 26, line 24 to page 27, line 6. A row address unit 306 together with the activation bits 304 (*e.g.*, for activating and deactivating rows) are used together with masked row-oriented load/store commands for controlling the movement of data between the DRAM array and an inactive register file. *See id.* at page 32, line 29 to page 33, line 3. As such, the wait time required to assemble data when a cache miss occurs is preempted, thereby improving the speed of program operation, as well as overall runtime efficiency. *See id.* at page 26, line 24 to page 27, line 6. Indeed, it is clear from the specification that a deactivate command is associated with controlling the movement of data (i.e., via data assembly unit 122) between register files and a DRAM array, and is issued based on a set of activation bits manipulated under program control.

In contrast, a "refresh command", which the Examiner has asserted as being the equivalence of the "deactivate command" recited by independent claim 1, is not associated with the movement of data. As will be appreciated by those skilled in the art, DRAM is a type of random access memory which stores data into capacitors within an integrated circuit. However, since in real world operation, capacitors leak charge over time, the information stored within the capacitors eventually fades. Accordingly, the capacitors (memory cells) of a DRAM array must be refreshed periodically in order to

maintain the data stored within the cells. Typically, a refresh operation is accomplished by reading the contents of a cell and then writing the contents of the cell back into the *same memory cell*, thereby replenishing the charge stored in the refreshed cell.

The Examiner's rationale for asserting a DRAM refresh operation as being analogous to the recited "deactivate command" appears to be based on the observation that during a refresh operation, the memory cell or cells being refreshed are inaccessible. *See* Office Action mailed 8/21/2007, page 5. In support of this position, the Examiner cited the Garcia reference as extrinsic evidence. The Garcia reference discloses that during a refresh cycle, DRAM memory cells may unable to respond to read or write commands requested by a scan test. *See* Garcia, col. 1, line 66 – col. 2, line 2. In other words, the Examiner appears to be suggesting that DRAM rows are "deactivated" during a refresh cycle because read or write operations cannot occur. However, Appellant asserts that the Examiner's logic is flawed. Rather than rendering the DRAM *unresponsive* to read or write operations, the refresh cycle itself is defined as including both a read and a write operation. For instance, as described above, a refresh operation typically constitutes reading the state of a memory cell, and writing the same state back into the same memory cell (i.e., to replenish capacitive charge). To the extent that the Examiner's reliance on the Garcia or Farmwald references has any basis whatsoever, it is that the refresh cycle must complete its own read and write operations (i.e., reading data from a memory cell and writing it back into the same memory cell) before subsequent read or write operations may be performed. It certainly does not mean the DRAM is "deactivated."

Furthermore, Appellant is unable to identify any teaching in either the Farmwald or Garcia references to suggest that the refresh command is associated with controlling the movement of data. Rather, as discussed above, a refresh command keeps or preserves data in its current location (i.e., reading data from a memory cell and writing it back into the same memory cell). Thus, contrary to the Examiner's assertions, Appellant submits

that when considered in view of the specification, one of ordinary skill in the art would not reasonably interpret a "deactivate command," as recited by independent claim 1, to be the equivalent of a DRAM refresh command.

In the Final Office Action, the Examiner acknowledged the foregoing distinctions but chose to maintain the rejection of independent claim 1 based on the Inagami, Luk, Farmwald, and Garcia references. Specifically, the Examiner stated:

> The examiner disagrees because the read/write being performed is merely for maintaining the contents of the DRAM. Said read/write is not a read/write included as part of the program code. During a refresh, program code cannot access the memory. Hence, the DRAM is deactivated from the viewpoint of program execution.
>
> Final Office Action, page 54.

As best understood from these statements, the Examiner is alleging that a refresh operation constitutes a deactivation command because a refresh operation is not part of the program code. Appellant respectfully disagrees with this line of reasoning. In particular, Appellant notes that the "deactivate command" is clearly recited by independent claim 1 as being part of the instruction set. As the Board will appreciate, in the context of computer programming, an instruction set is generally referred to as a set of commands or instructions (oftentimes defining the "code" of a program) that is executable by a processor. Thus, contrary to the Examiner's argument that the recited "deactivate" command and a DRAM refresh command are *not* part of the program code, independent claim 1 clearly recites that the deactivate command is part of the recited instruction set, and thus is part of the program code.

Accordingly, Appellant respectfully submits that the Examiner's reliance on the Farmwald and Garcia references as allegedly disclosing this recited feature is misplaced. Appellant further notes that the Examiner did not indicate that the Inagami or Luk

references, which were cited in combination with Farmwald and Garcia, were relied upon in this regard and, therefore, these additionally cited references fail to obviate the deficiencies the Farmwald and Garcia references. Thus, for at least this additional reason, Appellant respectfully submits that independent claim 1 is allowable over the combination of the Inagami, Luk, and Farmwald references, and that the Garcia references fails to provide evidence to overcome the deficiencies of these cited references.

4. Request Withdrawal of the Rejection.

In view of the deficiencies outlined above, among others, Appellant respectfully submits that the Examiner has failed to establish a *prima facie* case of obviousness with regard to independent claims 1 and 13 in view of the Inagami, Luk, Farmwald, and Garcia references. Further, claims 6-10, 12, and 14 are believed to be clearly patentable at least by virtue of their respective dependencies from either claims 1 or 13. Accordingly, Appellant respectfully requests that the Board direct the Examiner to withdraw the rejection of claims 1, 6-10, 12-14, and 13 under 35 U.S.C. § 103(a) and to allow these claims.

B. Second Ground of Rejection

The Examiner further rejected claims 23-25 under 35 U.S.C. § 103(a) as being unpatentable based on the combination of the Inagami and Luk references. Of these, claim 23 is independent. Appellant respectfully traverses this rejection.

1. The cited references, alone or in combination, fail to teach or suggest register files capable of loading or storing either an entire row of a DRAM array or selected columns of an entire row of a DRAM array either in a single operation or in response to a single latch signal, and further fail to teach or

suggest a command for doing the same, as recited by independent claim 23.

Independent claim 23 recites, *inter alia*, "first and second registers files, each of said register files capable of loading or storing an entire row of the DRAM array in response to a single latch signal" and "a command to load the entire row pointed to by said row address register into a selected set of registers of said register files in a single operation." (Emphasis added.) This feature is believed to be absent from the cited combination of the Inagami and Luk references. Moreover, Appellant notes the this recited feature of independent claim 23 is substantially similar to features recited by independent claims 1 and 13, which are addressed above with regard to the First Ground of Rejection in Section VII.A.2 of the present Appeal Brief. *See supra* Section VII.A.2. Thus, Appellant respectfully submits that independent claim 23 is allowable for at least the reasons discussed above with regard to independent claims 1 and 13.

In summary, the Examiner, in setting forth the rejection of independent claim 23 in the Final Office Action, relied solely upon the Inagami reference as allegedly disclosing a register file capable of loading or storing an entire row of a DRAM array in a single operation or in response to a single latch signal. *See* Final Office Action, pages 12-14. However, as discussed above, the DRAM array disclosed by the Inagami reference clearly requires at least four separate operations (and thus four separate latch signals) in order to load or store an entire row of a DRAM array to a register file. *See supra* Section VII.A.2. Thus, Appellant does not believe that the Inagami reference could be reasonably construed as disclosing the above-recited feature. Further, Appellant reemphasizes that the Examiner's attempt to interpret the term "entire row" as being either a single cell or a portion of a row in a memory array is not only in clear contradiction with the plain meaning of the term "entire," but also blatantly inconsistent with the interpretation that one skilled in the art would reach.

Moreover, Appellant notes that the Examiner did not indicate that the Luk reference, which was cited in combination with Inagami, was relied upon in this regard. As such, it is believed that the Luk reference fails to obviate the above-discussed deficiencies of the Inagami reference. For at least these reasons, Appellant respectfully submits that independent claim 23 is allowable over the combination of the Inagami and Luk references.

2. The cited references, alone or in combination, do not teach or suggest a dual-port register file including a first port operative to parallely transfer contents of a selected register file between a DRAM row and at least a second port operative to transfer data between the selected register file and a functional unit, as recited by independent claim 23.

Independent claim 23 further recites "first and second register files ... wherein said first and second register files comprise a parallel access port operative to parallely transfer contents of one of said register files between a DRAM row as selected by said row-address register, said first and second register files further comprising at least a second access port operative to transfer data between a selected one of said register files and said second functional unit." (Emphasis added.) Appellant respectfully submits that this feature is also absent from the Luk and Inagami references

Before addressing the deficiencies of the Examiner's rejection with regard to independent claim 23, Appellant directs the Board's attention to FIG. 2 of the present Application in which the dual port feature of the recited register files is clearly illustrated. *See* Application, FIG. 2. In particular, this figure illustrates an embodiment of the present invention in which the dual port feature is provided to the register file 112 by way of a series of switches 204 coupled to each of the data registers 0-k within the register file 112. *See id.* As shown in FIG. 2, each of the switches 204 includes a first port coupled to a parallel load/store channel and operative to transfer data to row 208 of a DRAM array (*e.g.*, via bus 216). *See* Application, page 28, lines 4-20; FIG. 2. Furthermore, each switch 204 also includes a second port coupling the register file to a functional unit 128

(through selector switch 206). *See id.* By way of this second access port, the selector switch 206 selectively couples the data registers 0-k of the register file 112 either to the functional units 128 or to the data assembly unit 122. *See id.* In other words, independent claim 23 clearly requires that a register file has two ports (*e.g.*, dual-port), wherein a first port is configured to transfer data between the register file and a row of a DRAM array and a second port is configured to transfer data between the register file and a functional unit, such that the dual port switches 204 are connect the register files to both the DRAM array and the functional unit. With these points in mind, Appellant respectfully submits that the Inagami reference (which appears to be the sole reference relied upon by the Examiner in the Final Office Action as allegedly teaching this recited feature) fails to disclose at least this recited feature of independent claim 23. *See* Final Office Action, pages 12-14.

In the present rejection, the Examiner, citing FIG. 1 of the Inagami reference, correlated one of the vector register files VR0-VR7 to the recited register file, one of the switches 120-123 as being analogous to both the recited "first port" and "second port" (thus allegedly providing the dual-port feature), the main storage structure 1 as being analogous to the recited DRAM array, and the load/store sub-pipes 2 as being analogous to the recited functional unit. *See id.* at page 13. Based on this interpretation, the Examiner alleged that the Inagami reference discloses a dual port register file having a first port operative to parallely transfer data between a vector register file (VR0-VR7) and a row of the asserted DRAM array, as well as a second port operative to transfer data between the vector register file (VR0-VR7) and the asserted functional unit. *See* Final Office Action, page 13; *see also* Inagami, FIG. 1. However, after carefully reviewing the passages and figures cited by the Examiner in the Final Office Action, Appellant is unable to ascertain how the Inagami reference teaches or suggests that a vector register file VR0 includes both a first port operative to parallely transfer data from the vector register file VR0 to a the asserted DRAM array (main storage 1) and a second port

operative to transfer data between the vector register file VR0 and the asserted functional unit (load/store sub-pipe 2).

To the contrary, it appears that the Inagami reference merely discloses that a functional unit (load/store sub-pipe 2) and a row in a DRAM array (main storage 1) are coupled in *series* to the vector register file VR0 by a single port, namely the switch 120. For example, referring to FIG. 1 of the Inagami reference, the main storage 1 is coupled to the load/store sub-pipe block 2 by the input/output connection lines 220-223 and 230-233. *See* Inagami, FIG. 1. Further, based on the Examiner's interpretation of the reference, the only connections between each of switches 120-123 (which the Examiner has correlated to each of the dual port switches 204 of the present invention) and the main storage 1 and load/store sub-pipe block 2 are respective single connection lines for each switch 120-123, designated in FIG. 1 of Inagami by reference numerals 240-243. Thus, even if the switches 120-123, which are associated with what the Examiner has identified as register files (VR0-VR7), could be properly correlated with the dual port switches 204 illustrated in FIG. 2 of the present Application, the switches 120-123 *clearly* do not show both a first port for transferring data between a vector register file and the main storage 1, *and* a second port for transferring data between vector register file and the load/store sub-pipe 2. In stark contrast, it appears that any data transferred to or from the asserted DRAM array (main storage 1) to the register files (VR0-VR7) must pass through the same access port connecting the functional unit (load/store sub-pipe 2) to the register files (VR0-VR7). *See id.* In other words, the Inagami reference only discloses a single access port. It does not, as the Examiner suggests, disclose two distinct ports including a first port for transferring data between a register file (*e.g.*, vector register file VR0) and a DRAM array (*e.g.*, main storage 1) and a second port for transferring data between a vector register file (*e.g.*, vector register file VR0) and a functional unit (*e.g.*, load/store sub-pipes 2), as recited by independent claim 23.

Moreover, Appellant notes that the Examiner previously acknowledged the lack of an explicit teaching of the recited first and second ports by the Inagami reference in the Office Action mailed August 21, 2007. For instance, the Examiner's rationale in asserting that a second port exists for transferring data is that "if transfers occur between DRAM and registers…then a port must couple registers to DRAM." Office Action mailed 8/21/2007, page 41. (Emphasis added.) This statement appears to be a modest attempt at an assertion that a second access port is inherently taught by the Inagami reference. As the Board will appreciate, if the Examiner purports to rely on a theory of inherency in rendering a claimed feature as anticipated, the extrinsic evidence must make clear that the missing descriptive matter is necessarily present in the thing described in the reference, and that it would be so recognized by persons of ordinary skill. *In re Robertson,* 169 F.3d 743, 49 U.S.P.Q.2d 1949 (Fed. Cir. 1999) (Emphasis added.) The mere fact that a certain thing *may* result from a given set of circumstances is not sufficient. *Id.*

In view of the controlling case law, Appellant submits to the Board that the Examiner's apparent assertion that a second port inherently exists merely because data transfer may or could occur between the register files (VR0-VR7) and the DRAM array (main storage 1) must necessarily fail when examined in view of the Inagami reference, for Inagami *clearly* shows that data transfer between the asserted register files (vector register files VR0-VR7) and one or more rows of the asserted DRAM array (main storage 1) and data transfer between the register files (VR0-VR7) and the asserted functional unit (load/store sub-pipe 2) may be accomplished using a single common port (*e.g.*, switch 120). Accordingly, the Examiner's contention that a second port inherently exists based upon the teachings of the Inagami reference cannot be supported.

In the Final Office Action, the Examiner appears to acknowledge this deficiency, but further asserted that the switch 273 shown in FIG. 1 of the Inagami reference, which is also coupled to the register file VR0, could be considered the recited second port. *See*

Final Office Action, page 52. Appellant respectfully disagrees. First, Appellant notes that the reference numeral 273 actually appears to refer to a signal bus connecting the operation sub-pipe block 5-3 and a matrix circuit 137. Upon further review, it appears that the Examiner intended to cite the switch 124 as the recited "second port" instead of the bus 273. Indeed, based on the close proximity of the reference numeral 273 to the switch 124 in FIG. 1 of the Inagami reference, it appears that the Examiner unintentionally cited the reference 273 in place of the reference number 124. Accordingly, for the purposes of the present discussion, Appellant assumes that the Examiner intended to assert the switch 124, which is coupled to the vector register file VR0 and the operation sub-pipe block 5-0, as being the recited "second port" of independent claim 23.

Appellant respectfully submits that even this modified interpretation of the Inagami reference fails to establish a *prima facie* case of obviousness with regard to independent claim 23. Particularly, even if the switch 120 and the switch 124 could be properly construed as the recited first and second ports, respectively, independent claim 23 requires not only that the recited register files include two ports, but that a first port is configured to transfer data between a register file and a DRAM array, and that a second port is configured to transfer data between a register file and a functional unit. However, as clearly shown in FIG. 1 of the Inagami reference, the switch 124 cited by the Examiner is configured to transfer data between the vector register VR0 to an operation sub-pipe 5. Based on the Examiner's interpretation of the Inagami reference, as discussed above, the operation sub-pipe 5 is neither the asserted DRAM array (*e.g.*, main storage 1) nor the asserted functional unit (*e.g.*, load/store sub-pipe 2). Thus, the Examiner's revised interpretation of the Inagami reference also fails to render independent claim 23 obviousness.

In view of the foregoing analysis, Appellant submits that the Inagami reference fails to teach or suggest, either explicitly, implicitly, or inherently, a register file having a

first port for transferring data between the register file and a DRAM array and a second port for transferring data between the register file and a functional unit, as recited by independent claim 23. Appellant further notes that the Examiner did not indicate that the Luk reference, which was cited in combination with the Inagami reference, was relied upon in this regard and, therefore, the Luk reference fails to obviate the deficiencies of the Inagami reference. Accordingly, Appellant respectfully submits that independent claim 23 is allowable over the combination of the Inagami and Luk references for at least this additional reason.

3. Request Withdrawal of the Rejection.

In view of the deficiencies outlined above, Appellant respectfully submits that the Examiner has failed to establish a *prima facie* case of obviousness with regard to independent claim 23 in view of the Inagami and Luk references. Further, claims 24 and 25 are believed to be clearly patentable at least by virtue of dependency from claim 23. Accordingly, Appellant respectfully requests that the Board direct the Examiner to withdraw the rejection of claims 23-25 under 35 U.S.C. § 103(a) and to allow these claims.

C. Third Ground of Rejection

The Examiner further rejected dependent claims 2-5 and 11 under 35 U.S.C. § 103(a) as being unpatentable based on the combination of the Inagami, Luk, Farmwald, and Parady references. Appellant respectfully traverses this rejection.

1. The cited references, alone or in combination, fail to disclose all the recited claim elements, and thus cannot establish a *prima facie* case of obviousness with regard to claims 2-5 and 11.

Appellant notes that claims 2-5 and 11 each depend from independent claim 1. As discussed above with regard to the First Ground of Rejection, independent claim 1 is believed to be to be allowable over the combination of the Inagami, Luk, and Farmwald

references. Specifically, Appellant notes that the Inagami, Luk, and Farmwald references, either alone or in combination, fail to teach or suggest: (1) the loading or storing of an entire row by a register file in a single operation or in response to a single latch signal; and (2) a command to deactivate a row after it has been previously precharged. *See supra* Sections VII.A.2 and VII.A.3.

In rejecting claims 2-5 and 11, the Examiner cited the Inagami, Luk, and Farmwald reference in further combination with the Parady reference. In particular, the Examiner relied on the Parady reference as allegedly disclosing register files capable of being places in active and inactive states. *See* Final Office Action, page 16. However, the Examiner did not indicate that the Parady reference was relied upon to obviate the above-discussed deficiencies of the Inagami, Luk, and Farmwald references with regard to independent claim 1. Accordingly, Appellant respectfully submits that claims 2-5 and 11 are clearly patentable at least by virtue of dependency from claim 1.

2. Request Withdrawal of the Rejection.

For at least the foregoing reasons, Appellant respectfully requests that the Board direct the Examiner to withdraw the rejection of claims 2-5 and 11 under 35 U.S.C. § 103(a) and to allow these claims.

D. Fourth Ground of Rejection

The Examiner further rejected dependent claims 26 and 27 under 35 U.S.C. § 103(a) as being unpatentable based on the combination of the Inagami, Luk, and Parady references. Appellant respectfully traverses this rejection.

1. The cited references, alone or in combination, fail to disclose all the recited claim elements, and thus cannot establish a *prima facie* case of obviousness with regard to claims 26 and 27.

Appellant notes that claims 26 and 27 each depend from independent claim 23. As discussed above with regard to the Second Ground of Rejection, independent claim 23 is believed to be to be allowable over the combination of the Inagami and Luk references. Specifically, Appellant notes that the Inagami and Luk references, either alone or in combination, fail to teach or suggest: (1) the loading or storing of an entire row by a register file in a single operation or in response to a single latch signal; and (2) a register file having a first port for transferring data between the register file and a DRAM array and a second port for transferring data between the register file and a functional unit. *See supra* Sections VII.B.1 and VII.B.2.

In the rejection of claims 26 and 27, the Examiner cited the Inagami and Luk references in further combination with the Parady reference. In particular, the Examiner relied on the Parady reference as allegedly disclosing register files capable of being placed in active and inactive states. *See* Final Office Action, pages 21-22. However, the Examiner did not indicate that the Parady reference was relied upon to obviate the above-discussed deficiencies of the Inagami and Luk references with regard to independent claim 23. Accordingly, Appellant respectfully submits that claims 26 and 27 are clearly patentable at least by virtue of dependency from claim 23.

2. Request Withdrawal of the Rejection.

For at least the foregoing reasons, Appellant respectfully requests that the Board direct the Examiner to withdraw the rejection of claims 26 and 27 under 35 U.S.C. § 103(a) and to allow these claims.

E. Fifth Ground of Rejection

The Examiner rejected claims 16-22, 28-32, 34-43, 45, and 49-50 under 35 U.S.C. § 103(a) as being unpatentable based on the combination of the Inagami, Luk, Parady,

and Bissett references. Of these, claims 16, 28, 45, 49, and 50 are independent. Appellant respectfully traverses this rejection.

1. **The cited references, alone or in combination, fail to teach or suggest register files capable of loading or storing either an entire row of a DRAM array or selected columns of an entire row of a DRAM array either in a single operation or in response to a single latch signal, and further fail to teach or suggest a command for doing the same, as recited by independent claims 16, 28, 49, and 50.**

Independent claim 16 recites, *inter alia*, "first and second register files, each of said register files having a plurality of data registers capable of loading or storing an entire row of the DRAM array in response to a single latch signal" and "a command to load a set of selected elements of the row … into a selected set of said data registers, said selection of elements … wherein all the selected elements of the row are loaded into the selected sets of data registers in a single operation." (Emphasis added.) Independent claim 28 recites, *inter alia*, "first and second dual-port registers files, each of said register files capable of parallely transferring data between an entire row of said DRAM array in a single operation" and a command to "transfer data between an entire row of said DRAM array and a selected inactive data register file, wherein the transfer occurs in a single operation." (Emphasis added.) Independent claim 49 recites a method for use in an "embedded DRAM array having … first and second dual-port registers files each capable of (i) parallel transfer of data between an entire row of said embedded DRAM array in a single operation," wherein the method comprises "transferring data between an entire row of said embedded DRAM array and a selected inactive data register file, wherein the transfer of the data occurs in a single operation." (Emphasis added.) Independent claim 50 recites, *inter alia*, "embedded first and second registers files, each of said register files capable of loading or storing an entire row of the DRAM array in response to a single latch signal" and "a command to load the entire row pointed to by said row address register into a selected set of registers of said register files … wherein loading the entire row is performed in a single operation." (Emphasis added.) At least these recited

features are believed to be absent from the cited combination of the Inagami, Luk, Parady, and Bissett references. Moreover, Appellant notes the these recited features of independent claims 16, 28, 49, and 50 are substantially similar to features recited by independent claims 1 and 13, which are addressed above with regard to the First Ground of Rejection in Section VII.A.2 of the present Appeal Brief. *See supra* Section VII.A.2. Thus, Appellant respectfully submits that independent claims 16, 28, 49, and 50 are allowable for at least the reasons discussed above with regard to independent claims 1 and 13.

In summary, the Examiner, in setting forth the rejection of independent claims 16, 28, 49, and 50, relied solely upon the Inagami reference as allegedly disclosing a register file capable of loading or storing an entire row of a DRAM array in a single operation or in response to a single latch signal. *See* Final Office Action, pages 23-24, 28-30, 42-43, 45-47. However, as discussed above, the DRAM array disclosed by the Inagami reference clearly requires at least four separate operations (and thus four separate latch signals) in order to load or store an entire row of a DRAM array to a register file. *See supra* Section VII.A.2. Thus, Appellant does not believe that the Inagami reference could be reasonably construed as disclosing the above-recited feature. Further, Appellant reemphasizes that the Examiner's attempt to interpret the term "entire row" as being either a single cell or a portion of a row in a memory array is not only in clear contradiction with the plain meaning of the term "entire," but also blatantly inconsistent with the interpretation that one skilled in the art would reach.

Moreover, Appellant notes that the Examiner did not indicate that the Luk, Parady, or Bissett references, which were cited in combination with the Inagami reference, were relied upon in this regard. As such, Appellant asserts that none of these additionally cited references obviate the above-discussed deficiencies of the Inagami reference. For at least these reasons, Appellant respectfully submits that independent

claims 16, 28, 49, and 50 are allowable over the combination of the Inagami, Luk, Parady, and Bissett references.

2. The cited references, alone or in combination, do not teach or suggest a dual-port register file including a first access port operative to parallely transfer contents of a selected register file between a DRAM row and at least a second access port operative to transfer data between the selected register file and a functional unit, as recited by independent claim 50.

Independent claim 50 recites, *inter alia*, "first and second register files ... wherein said first and second registers files comprise a parallel access port operative to parallely transfer contents of said register file between a DRAM row as selected by said row-address register, each of said register file further comprising at least a second access port operative to transfer data between a selected register file and said second functional unit." (Emphasis added.) These recited features are believed to be absent from the cited combination of the Inagami, Luk, Parady, and Bissett references. Moreover, Appellant notes the these recited features of independent claim 50 are substantially similar to features recited by independent claim 23, which are addressed above with regard to the Second Ground of Rejection in Section VII.B.2 of the present Appeal Brief. *See supra* Section VII.B.2. Thus, Appellant respectfully submits that independent claim 50 is allowable for at least the reasons discussed above with regard to independent claim 23.

In summary, the Examiner, in setting forth the rejection of independent claim 50, relied solely upon the Inagami reference as allegedly disclosing a register file having a first access port operative to transfer data between the register file and a DRAM array and a second access port operative to transfer data between the register file and a functional unit. *See* Final Office Action, page 46. However, as discussed above, the Inagami reference does not appear to disclose a first access port operative to transfer data between a register file and a DRAM array and a second access port operative to transfer data between the register file and a functional unit. In particular, the switch 120, which the Examiner asserted as being the recited first access port, appears to be *directly* coupled to

the load/store sub-pipe 2, which the Examiner asserted as the recited functional unit. Moreover to the extent that data from the main storage 1, which the Examiner asserted as the recited DRAM array, may be transferred to the vector register file VR0 (the asserted register file) through the load/store sub-pipe 2 and the switch 120, Appellant notes that the transfer of data between the vector register file VR0 and the main storage 1 (asserted DRAM array) or the load/store sub-pipe 2 (asserted functional unit) would occur through the *same* port, namely the switch 120. The data transfers would not, as recited by independent claim 50, occur through respective first and second access ports. Additionally, Appellant reemphasizes that the Examiner's reliance on the switch 124 (cited as switch 273 in the Final Office Action) fails to overcome the foregoing deficiencies because the switch 124 appears to facilitate the transfer of data between the vector register VR0 and the operation sub-pipe 5, which is neither a DRAM array nor a functional unit based upon the Examiner's interpretation of the Inagami reference.

Additionally, Appellant notes that the Examiner did not indicate that the Luk, Parady, or Bissett references were relied upon to obviate these deficiencies of the Inagami reference. As such, Appellant submits that independent claim 50 is allowable over the combination of the Inagami, Luk, Parady, and Bissett references.

3. The cited references, alone or in combination, do not teach or suggest a dual-port register file having a parallel access port coupled to a DRAM array, as recited by independent claim 45.

Independent claim 45 generally recites "first and second dual-port registers files, whereby the first port of each of said register files is a parallel access port and is parallely coupled to said DRAM array." (Emphasis added.) This feature is believed to be absent from the combination of the Inagami, Luk, Parady, and Bissett references.

As discussed above with regard to the rejection of independent claim 23 under the Second Ground of Rejection, the elements in the Inagami reference that the Examiner cited as being a "port" of a register file (*e.g.*, vector register file VR0) are the switch 120

and the switch 124. *See supra* Section VII.B.2. As shown in FIG. 1 of the Inagami reference, the switch 120 is coupled to the vector register file VR0 and the load/store sub-pipe 2, which the Examiner asserted as being the recited second functional unit. *See* Inagami, FIG. 1. Further, the switch 124 is coupled to the vector register file VR0 and the operation sub-pipe 5. *See id.* In other words, even if either of the switches 120 or 124 could reasonably be characterized as being analogous to the recited "first access port," Appellant notes that neither of the switches 120 or 124 are coupled to the main storage 1, which the Examiner asserted as being the recited DRAM array, as recited by independent claim 45. Accordingly, Appellant submits that no *prima facie* case of obviousness has been established based on the combination of the Inagami, Luk, Parady, and Bissett references with regard to at least this recited feature of independent claim 45.

4. The cited references, alone or in combination, do not teach or suggest register files capable of being placed into an active and an inactive state, nor does the prior art teach or suggest commands for manipulating active and inactive register files, as recited by independent claims 16, 28, 45, and 49.

Independent claim 16 recites, *inter alia*, "first and second register files … each of said register files also being capable of being placed into an active state and an inactive state." (Emphasis added.) Independent claim 28 recites, *inter alia*, "first and second dual-port registers files, each of said register files … also being capable of being placed into an active state and an inactive state" and "a command to place said selected inactive data register file into said active state." (Emphasis added.) Independent claim 45 recites, *inter alia*, "first and second dual-port registers files, whereby the first port of each of said register flies is a parallel access port and is parallely coupled to said DRAM array, each or said register files being capable of being placed into an active state and an inactive state" and "a command that causes said register file in the inactive state to assume the active state and said register file in the active state to assume the inactive state." (Emphasis added.) Independent claim 49 recites a method comprising, *inter alia*, "manipulating data in a data register within a register file that is in an the active state" and "unidirectionally

transferring data between an entire row of said embedded DRAM array and a selected inactive data register file." Thus, each of independent claims 16, 28, 45, and 49 recite register files capable of being placed in an active and inactive state. Also, independent claims 28 and 45 additionally recite commands for placing an inactive register into an active state. These feature are believed to be absent from the cited combination of the Inagami, Luk, Parady, and Bissett references.

In the present Application, the active and inactive register files are described in the context of an intelligent cache structure that splits the embedded-DRAM architecture into a core execution portion and an intelligent data assembly portion. *See* Application, page 6, lines 6-21; page 24, line 6 to page 25, line 3. In order to implement intelligent caching, a data assembly unit 122 assembles data in inactive register files. *See id.* at page 6, lines 13-16. Assembling the data may involve performing register-to-register move operations, as well as load/store operations. *See id.* at page 24, lines 27-29. Once the data is *finished being assembled* by the data assembly unit 122, the register file switches from an inactive to an active state. *See id.* at page 6, lines 13-16. Further, once a register is switched to an active state, it functions as an architectural register, being visible to the program executed by functional units. *See id.* Indeed, as *clearly* set forth in the specification, active and inactive states are determined based on whether or not the data assembly unit (*e.g.*, 122) has finished assembling the required data. Additionally, the intelligent caching scheme described in the present Application is an improvement over the prior art hit-or-miss caching schemes. *See id.* at page 25, lines 16-20.

As discussed above, it is well established legal precedent that pending claims must be given an interpretation that is reasonable and consistent with the *specification*. See *In re Prater*, 415 F.2d 1393, 1404-05, 162 U.S.P.Q. 541, 550-51 (C.C.P.A. 1969) (Emphasis added); *see also In re Morris,* 127 F.3d 1048, 1054-55, 44 U.S.P.Q.2d 1023, 1027-28 (Fed. Cir. 1997). In rejecting independent claims 16, 28, 45, and 49, the Examiner acknowledged that the Inagami reference fails to disclose that the vector

registers (VR0-VR7) are capable of being placed into the recited active and inactive states. However, the Examiner attempted to fill this deficiency based on the teachings set forth in the Parady reference

The Parady reference generally describes a method for program thread switching. Specifically, the Examiner asserted that the Parady reference teaches a switch command for toggling between active and inactive states based on which program thread is currently active. *See* Final Office Action, pages 23-24, 29-30, 39, 42-43. However, Appellant asserts that this interpretation of the terms "active" and "inactive" is not reasonable and consistent with the Appellant's specification, as discussed above, which *clearly* states that "active" or "inactive" register files are determined based on whether or not a data assembly unit has completed assembly of data within the register file. Furthermore, as discussed above, the architecture set forth in present Application utilizes intelligent caching, which is an improvement over the prior art hit-or-miss caching schemes described in the Parady reference. For example, the Parady reference discloses that "[t]he indication that a thread switch is required is provided on a line 114 providing an L2 miss indication from cache control/system interface 22." Parady, col. 3, lines 56-65. (Emphasis added.) Indeed, the Parady reference clearly shows that toggling between register files is based on a thread switch, not the assembling of data within a register file, as Appellant's submit would be the proper interpretation based on the teachings conveyed in Appellant's disclosure. As such, Appellant asserts that the Examiner's interpretation of the terms "active" and "inactive" are not reasonable and consistent with the specification. Therefore, Appellant does not believe that the Parady reference can be relied upon as teaching for suggesting that register files can be placed into active and inactive states.

The Examiner further rejected claims 28 and 45, stating that the Inagami reference, when considered in combination with the Parady reference in the manner discussed above, discloses a command for placing an inactive register into an active state.

However, in view of the foregoing discussion regarding the interpretation of the terms "active" and "inactive," Appellant submits that the Examiner's rejection fails to consider the meaning of these terms when interpreted in view of the specification. Accordingly, Appellant does not believe that the Parady reference teaches or suggests a command for transitioning register files between "active" and "inactive" states, which is defined in the specification as being based upon the assembling of data by a data assembly unit.

In view of the foregoing analysis, Appellant submits that the Parady reference cannot properly be relied upon as teaching the recited "active" and "inactive" register states, as generally recited by independent claims 16, 28, 45, and 49. Appellant further notes that the Examiner did not indicate that the Luk or Bissett references, which were cited in combination with the Inagami and Parady references, were relied upon in this manner. As such, neither the Luk nor Bissett references obviate the deficiencies the Inagami and Parady references. Accordingly, Appellant respectfully submits that independent claims 16, 28, 45, and 49 are allowable over the combination of the Inagami, Luk, Parady, and Bissett references for at least this additional reason.

5. Request Withdrawal of the Rejection.

In view of the deficiencies outlined above, Appellant respectfully submits that the Examiner has failed to establish a *prima facie* case of obviousness with regard to independent claims 16, 28, 45, 49, and 50 based upon the combination of the Inagami, Luk, Parady, and Bissett references. Further, claims 17-22, 29-32, and 34-43 are believed to be clearly patentable at least by virtue of their respective dependencies from claims 16 or 28. Accordingly, Appellant respectfully requests that the Board direct the Examiner to withdraw the rejection of claims 16-22, 28-32, 34-43, 45, and 49-50 under 35 U.S.C. § 103(a) and to allow these claims.

F. Sixth Ground of Rejection

The Examiner further rejected dependent claim 44 under 35 U.S.C. § 103(a) as being unpatentable based on the combination of the Inagami, Luk, Parady, Bissett, and Farmwald references. Appellant respectfully traverses this rejection.

1. The cited references, alone or in combination, fail to disclose all the recited claim elements, and thus cannot establish a *prima facie* case of obviousness with regard to claim 44.

Claim 44 depends from independent claim 28. As discussed above with regard to the Fifth Ground of Rejection, independent claim 28 is believed to be to be allowable over the combination of the Inagami, Luk, Parady, and Bissett references. Specifically, Appellant notes that the Inagami, Luk, Parady, and Bissett references, either alone or in combination, fail to teach or suggest at least the following features recited by independent claim 28: (1) a register file capable of loading or storing an entire row of a DRAM array in a single operation or in response to a single latch signal; (2) register files capable of being placed in an inactive and active states; and (3) a command for causing a register file to transition between the inactive and active state. *See supra* Sections VII.E.1 and VII.E.4.

In the rejection of claim 44, the Examiner cited the Inagami, Luk, Parady, and Bissett references in further combination with the Farmwald reference. In particular, the Examiner relied on the Farmwald reference as allegedly disclosing speculative precharging of a row in a DRAM array. *See* Final Office Action, page 48. However, the Examiner did not indicate that the Farmwald reference was relied upon to obviate the above-discussed deficiencies of the Inagami, Luk, Parady, and Bissett references with regard to independent claim 28. Accordingly, Appellant respectfully submits that claim 44 is clearly patentable at least by virtue of dependency from claim 28.

2. Request Withdrawal of the Rejection.

For at least the foregoing reasons, Appellant respectfully requests that the Board direct the Examiner to withdraw the rejection of claim 44 under 35 U.S.C. § 103(a) and to allow this claim.

G. Seventh Ground of Rejection

The Examiner further rejected independent claim 46 under 35 U.S.C. § 103(a) as being unpatentable based on the combination of the Inagami, Luk, and Krishnamohan references. Appellant respectfully traverses this rejection.

1. The cited references, alone or in combination, fail to teach or suggest loading a plurality of words designated by a row address register into designated sets of data registers (e.g., a register file) in a single operation, as recited by independent claim 46.

Independent claim 46 recites a method comprising, *inter alia*, "loading a plurality of words of a row designated by said at least one row address register into designated sets of data registers in a single operation." (Emphasis added.) In other words, claim 46 requires that a row specified in by a row address is loaded into a register file in a single operation. This feature is believed to be absent from the cited combination of the Inagami, Luk, and Krishnamohan references. Moreover, Appellant notes the this recited feature of independent claim 46 is similar to features recited by independent claims 1 and 13, which are addressed above with regard to the First Ground of Rejection in Section VII.A.2 of the present Appeal Brief. *See supra* Section VII.A.2. As such, Appellant respectfully submits that independent claim 46 is allowable for at least the reasons discussed above with regard to independent claims 1 and 13.

In summary, the Examiner, in setting forth the rejection of independent claim 46 relied solely upon the Inagami reference as allegedly disclosing a the loading of a row

specified by a row address stored in a row address register into a register file (*e.g.*, designated sets of data registers) in a single operation. *See* Final Office Action, pages 50-51. However, as discussed above, the DRAM array disclosed by the Inagami reference clearly requires at least four separate operations (and thus four separate latch signals) in order to load a row of a DRAM array to a register file. *See supra* Section VII.A.2. Thus, Appellant does not believe that the Inagami reference could be reasonably construed as disclosing the above-recited feature. Further, Appellant reemphasizes that the Examiner's attempt to interpret the term "row" as being a subset of a row (*e.g.*, a single cell) within a memory array is in clear contradiction with the plain meaning of the term "row" that one skilled in the art would understand when applied in the context of memory devices. *See id.*

Still further, Appellant notes that the Examiner did not indicate that the Luk or Krishnamohan references, which were cited in combination with Inagami, were relied upon in this regard. For instance, the Examiner only relied upon the Luk reference solely for the teaching that a processor and a DRAM array may be embedded on a single chip, and further relied on the Krishnamohan reference as allegedly teaching speculative precharging based upon historical program execution data. *See* Final Office Action, pages 49-50. Accordingly, Appellant submits that these additionally cited references fail to obviate the above-discussed deficiencies of the Inagami reference. For at least these reasons, it is believed that independent claim 46 is allowable over the combination of the Inagami, Luk, and Krishnamohan references.

2. Request Withdrawal of the Rejection.

For at least the foregoing reasons, Appellant respectfully requests that the Board direct the Examiner to withdraw the rejection of independent claim 46 under 35 U.S.C. § 103(a) and to allow this claim.

H. Eighth Ground of Rejection

Lastly, the Examiner rejected dependent claim 47 under 35 U.S.C. § 103(a) as being unpatentable based on the combination of Inagami, Luk, Krishnamohan, and Farmwald, with Garcia cited as extrinsic evidence with regard to DRAM refresh cycles. Appellant respectfully traverses this rejection.

1. The cited references, alone or in combination, fail to disclose all the recited claim elements, and thus cannot establish a *prima facie* case of obviousness with regard to claim 47.

Claim 47 depends from independent claim 46. As discussed above with regard to the Seventh Ground of Rejection, independent claim 46 is believed to be to be allowable over the combination of the Inagami, Luk, and Krishnamohan references. Specifically, Appellant notes that the Inagami, Luk, and Krishnamohan references, either alone or in combination, fail to teach or suggest the loading of a row specified by a row address stored in a row address register into a register file (*e.g.*, designated sets of data registers) in a single operation. *See supra* Section VII.G.1.

In the rejection of claim 47, the Examiner cited the Inagami, Luk, and Krishnamohan references in further combination with the Farmwald reference. The Examiner also cited the Garcia reference as extrinsic evidence with regard to the refreshing of DRAM cells. In particular, the Examiner relied on the Farmwald and Garcia references as allegedly disclosing a deactivation command to deactivate a precharged row of a DRAM array. *See* Final Office Action, page 51. However, the Examiner did not indicate that the Farmwald or Garcia references were relied upon to obviate the above-discussed deficiencies of the Inagami, Luk, and Krishnamohan references with regard to independent claim 46. Accordingly, Appellant respectfully submits that claim 47 is clearly patentable at least by virtue of dependency from claim 46.

2. Request Withdrawal of the Rejection.

For at least the foregoing reasons, Appellant respectfully requests that the Board direct the Examiner to withdraw the rejection of claim 47 under 35 U.S.C. § 103(a) and to allow these claims.

Conclusion

Appellant respectfully submits that all pending claims are in condition for allowance. However, if the Examiner or the Board wishes to resolve any other issues by way of a telephone conference, the Examiner or Board is kindly invited to contact the undersigned attorney at the telephone number indicated below.

Respectfully submitted,

Date: October 22, 2008

/Robert A. Manware/

Robert A. Manware
Reg. No. 48,758
FLETCHER YODER
7915 FM 1960 West, Suite 330
Houston, TX 77070
(281) 970-4545

VIII. **APPENDIX OF CLAIMS ON APPEAL**

1. An embedded-DRAM (dynamic random access memory) processor comprising:

an embedded DRAM array comprising a plurality of random access memory cells arranged in rows and columns;

a row address register that holds a pointer that points to a row of the DRAM array;

one or more sets of registers, each of said sets of registers capable of loading or storing an entire row of the DRAM array in response to a single latch signal; and

an instruction set which includes:

(i) at least one command to perform an arithmetic operation on said row address register;

(ii) a command to precharge (activate) the row pointed to by said row address register;

(iii) a command to deactivate said row pointed to by said row address register after it had been precharged by the command to precharge;

(iv) a command to load a plurality of words of the precharged row into designated sets of data registers; and

(v) a command to load selected columns of the precharged row into designated sets of data registers, said selection based on bits in a mask, wherein all the selected columns of the precharged row are loaded into the designated sets of data registers in a single operation.

2. The embedded-DRAM processor according to Claim 1, further comprising:

first and second sets of functional units, said first and second sets of functional units having respective first and second instruction sets and capable of accessing first and second sets of said registers;

a command to select one of said first and second sets of registers to be an architectural set of registers accessible to said first set of functional units;

a command to deselect the other of said first and second sets of registers so that it is no longer an architectural register set accessible to said first set of functional units;

a command to select one of said first and second sets of registers to be an architectural set of registers accessible to said second set of functional units; and

a command to deselect the other one of said first and second sets of registers so that it is no longer an architectural register set accessible to said second set of functional units.

3. The embedded-DRAM processor according to Claim 1, further comprising:

first and second sets of functional units, said first and second sets of functional units having respective first and second instruction sets and accessing first and second sets of registers; and

a command which selects one of said first and second sets of registers to be an architectural set of registers accessible to said first set of functional units, and, at the same time, which deselects the other one of said one of said first and second sets of registers to be an architectural set of registers accessible to said second set of functional units.

4. The embedded-DRAM processor according to Claim 1, further comprising:

first and second sets of functional units, said first and second sets of functional units having respective first and second instruction sets and accessing first and second ones of said register sets; and

whereby said first and second instruction sets are subsets of said instruction set of said embedded-DRAM processor.

5. The embedded-DRAM processor according to Claim 4, whereby said second set of functional units comprises a functional unit that is a multi-issue functional unit, and further comprises:

a dispatch unit;

a plurality of functional units which each execute a respective instruction stream as dispatched by said dispatch unit.

6. The embedded-DRAM processor according to Claim 1, further comprising a plurality of DRAM arrays.

7. The embedded-DRAM processor according to Claim 1, further comprising:

at least one functional unit;

whereby said one or more sets of registers comprise a plurality of register files, each of said register files comprising a parallel access port operative to load or store contents of said register file in a single cycle from or to a DRAM row as selected by said row-address register, each of said register files further comprising at least a second access port operative to transfer data between said functional unit and a selected subset register in said register file.

8. The embedded-DRAM processor according to Claim 7, further comprising:

a second functional unit;

whereby said first functional unit executes a first command to perform logical processing on the contents of one or more registers within a selected active one of said register sets, and said second functional unit executes a second command to parallely transfer data between a selected inactive one of said register sets and said DRAM array.

9. The embedded-DRAM processor according to Claim 8, wherein said first and second functional units execute said first and second commands substantially contemporaneously.

10. The embedded-DRAM processor according to Claim 8, further comprising:

a fist software module comprising a set of data manipulation commands, said first software module executed by said first functional unit; and

a second software module comprising a set of parallel data transfer commands, said second software module being executed by said second functional unit;

whereby said second software module operates in support of said first software module to prefetch data from said DRAM array into one of said register files in advance of said data being needed by said first software module.

11. The embedded-DRAM processor according to Claim 10, wherein:

said first software module contains an instruction that reference registers within an architectural register set visible to said first functional unit, whereby said architectural register set corresponds at least partially to said one of said register files that is in an active state;

said second software module contains instructions that cause data to be transferred between an inactive register set and said DRAM array, and second software module also executes a command to toggle a selected register set between said active and inactive states.

12. The embedded-DRAM processor according to Claim 1, further comprising:

first and second sets of functional units, said first and second sets of functional units having respective first and second instruction subsets;

whereby said first instruction subset includes said command (i) and the second instruction subset includes said commands (ii), (iii), (iv) and (v).

13. An embedded-DRAM (dynamic random access memory) processor comprising:

an embedded DRAM array comprising a plurality of random access memory cells arranged in rows and columns;

a row address register that holds a pointer that points to a row of the DRAM array;

one or more sets of data registers, each of said sets of data registers capable of loading or storing an entire row of the DRAM array in response to a single latch signal;

a bit mask to select one or more data locations within at least one of said register sets; and

an instruction set which comprises at least:

(i) a command to perform arithmetic operations on said row address register;

(ii) a command to precharge (activate) the row pointed to by said row address register;

(iii) a command to load a set of selected elements of the precharged row pointed to by said row address register into a selected set of said data registers, said selection of elements based on bits in said bit mask, wherein all the selected elements of the precharged row are loaded into the selected sets of data registers in a single operation;

wherein the command to precharge is executed to precharge the row prior to the command to load so that at the time the command to load is issued, the command to load can execute without the need to wait for the row to precharge.

14. The embedded-DRAM processor according to Claim 13, wherein said load command causes an entire row that was previously precharged to be loaded.

16. An embedded-DRAM (dynamic random access memory) processor comprising:

an embedded DRAM array comprising a plurality of random access memory cells arranged in rows and columns;

a row address register that holds a pointer that points to a row of the DRAM array;

first and second register files, each of said register files having a plurality of data registers capable of loading or storing an entire row of the DRAM array in response to a single latch signal, each of said register files also being capable of being placed into an active state and an inactive state;

a set of functional units that perform logical operations on data accessed from a set of architectural registers, wherein registers placed into the active state appear as architectural registers to a set of functional units, and registers in the inactive state are not accessible by the functional units;

a bit mask to select one or more locations within at least one of said register files; and

an instruction set which comprises at least:

(i) a command to perform an arithmetic operation on said row address register; and

(ii) a command to load a set of selected elements of the row pointed to by said row address register into a selected set of said data registers, said selection of elements based on bits in said bit mask, wherein all the selected elements of the row are loaded into the selected sets of data registers in a single operation, the selected set of said data registers being in the inactive state.

17. The embedded-DRAM processor of Claim 16, wherein the instruction set further comprises:

(iii) a command to toggle a register set between said active and inactive states.

18. The embedded-DRAM processor of Claim 17, wherein said toggle command causes said first register tile to toggle from the inactive state to the active state and also causes the second register file to toggle from the active state to the inactive state.

19. The embedded-DRAM processor of Claim 16, wherein the instruction set further comprises:

(iii) a command to manipulate the bits in the bit mask.

20. The embedded-DRAM processor according to Claim 16, further comprising:

first and second sets of functional units;

first and second sets of instructions capable of accessing said first and second register sets; and

said instruction set further comprises at least:

(iii) a command to select one of said first and second sets of registers to be an architectural set of registers accessible to said first set of functional units; and

(iv) a command to select one of said first and second sets of registers to be an architectural set of registers accessible to said second set of functional units.

21. The embedded-DRAM processor according to Claim 20, said instruction set further comprising:

(v) a command to deselect the other of said first and second sets of registers so that it is no longer an architectural register set accessible to said first set of functional units; and

(vi) a command to deselect the other one of said first and second sets of registers so that it is no longer an architectural register set accessible to said second set of functional units.

22. The embedded-DRAM processor according to Claim 20, wherein at least one of said sets of functional units contains a single functional unit.

23. An embedded-DRAM (dynamic random access memory) processor comprising:

an embedded DRAM array comprising a plurality of random access memory cells arranged in rows and columns;

a row address register that holds a pointer that points to a row of the DRAM array;

first and second registers files, each of said register files capable of loading or storing an entire row of the DRAM array in response to a single latch signal; and

an embedded processor comprising first and second functional units, said first and second functional units having respective first and second instruction sets and capable of accessing said first and second register files;

wherein said first and second register files comprise a parallel access port operative to parallely transfer contents of one of said register files between a DRAM row as selected by said row-address register, said first and second register files further comprising at least a second access port operative to transfer data between a selected one of said register files and said second functional unit;

wherein said first instruction set comprises at least:

(i) a command to manipulate data in a data register within a register file; and

wherein said second instruction set comprises at least:

(ii) a command to perform an arithmetic operation on said row address register;

(iii) a command to load the entire row pointed to by said row address register into a selected set of registers of said register files in a single operation.

24. The embedded-DRAM processor according to Claim 23, wherein said first and second functional units each respectively execute a command from said first and second instruction sets substantially contemporaneously.

25. The embedded-DRAM processor according to Claim 24, further comprising:

a first software module comprising data manipulation commands drawn from said first instruction set, said first software module executed by said first functional unit; and

a second software module comprising a parallel data transfer command drawn from said second instruction set, said second software module being executed by said second functional unit;

whereby said second software module operates in support of said first software module to prefetch data from said DRAM array into one of said register files in advance of said data being needed by said first software module.

26. The embedded-DRAM processor of Claim 23, wherein the second instruction set further comprises

(iv) a command to toggle a register set between an active state and an inactive state.

27. The embedded-DRAM processor of Claim 26, wherein said toggle command causes said first register file to toggle from the inactive state to the active state and also causes the second register file to toggle from the active state to the inactive state.

28. An embedded-DRAM (dynamic random access memory) processor comprising:

an embedded DRAM array comprising a plurality of random access memory cells arranged in rows and columns;

first and second dual-port registers files, each of said register files capable of parallely transferring data between an entire row of said DRAM array in a single operation, each of said register files also being capable of being placed into an active state and an inactive state; and

first and second embedded functional units, said first and second functional units having respective first and second instruction sets that operate on registers of an architectural register set, said architectural register set comprising one of the first and second dual-port registers files that is currently in the active state;

wherein said first instruction set comprises at least:

(i) a command to manipulate data in a data register within a register file; and

wherein said second instruction set comprises at least:

(ii) a command to unidirectionally transfer data between an entire row of said DRAM array and a selected inactive data register file, wherein the transfer occurs in a single operation;

(iii) a command to place said selected inactive data register file into said active state, wherein when the inactive register file is activated, it becomes an architectural register set of said first functional unit.

29. The embedded-DRAM processor of Claim 28, wherein said command to unidirectionally transfer data causes data to be transferred from a row of the DRAM array to said selected inactive data register file.

30. The embedded-DRAM processor of Claim 28, wherein said command to unidirectionally transfer data causes data to be transferred from said selected inactive data register file to a row of the DRAM array.

31. The embedded-DRAM processor of Claim 28, wherein said command to place the selected inactive register file into the active state is a command that also causes the remaining register file to toggle from the active state into the inactive state.

32. The embedded-DRAM processor of Claim 28, further comprising:

at least one additional register file;

whereby said command to place the selected inactive register file into the active state is a command that also causes a selected other register file to toggle from the active state into the inactive state.

34. The embedded-DRAM processor of Claim 28, further comprising:

at least one bit mask; and

the second instruction set further comprises:

(iv) a command to move a subset of elements between a selected register file and a selected row of said DRAM array, whereby said subset is identified by said bit mask.

35. The embedded-DRAM processor according to Claim 28, wherein said first functional unit is a multi-issue functional unit and further comprises:

a dispatch unit;

a plurality of functional units that each execute a respective instruction stream as dispatched by said dispatch unit.

36. The embedded-DRAM processor according to Claim 28, further comprising:

a first software module comprising a set of data manipulation commands drawn from said first instruction set, said first software module executed by said first functional unit; and

a second software module comprising a set of parallel data transfer commands drawn from said second instruction set, said second software module being executed by said second functional unit;

wherein said second software module operates in support of said first software module to prefetch data from said DRAM array into one of said register tiles in advance of said data being needed by said first software module.

37. The embedded-DRAM processor according to Claim 28, wherein:

said first software module contains an instruction that references registers within an architectural register set visible to said first functional unit, whereby said architectural register set corresponds at least partially to said one of said register files that is in an active state;

said second software module contains instructions that cause data to be transferred between an inactive register set and said DRAM array, and second software module also executes a command to toggle a selected register set between said active and inactive states.

38. The embedded-DRAM processor according to Claim 28, whereby each of said register files contain a number of words, N, matched to the number of words in of a row of said DRAM array, and said unidirectional transfer comprises moving said selected row in its entirety to said selected register file.

39. The embedded-DRAM processor according to Claim 28, further comprising:

a mask and switch unit interposed between said DRAM array and at least one of said register files.

40. The embedded-DRAM processor according to Claim 28, wherein said second set of instructions comprises:

a command to cause data to be moved from one register to another within a given one of said register files.

41. The embedded-DRAM processor according to Claim 28, wherein said second instruction set is used to implement an intelligent caching scheme, whereby said register files act as a cache and said second set of instructions are executed in lieu of a standard cache that maintains most recently used data and enforces a set associative or a direct-mapped caching policy.

42. The embedded-DRAM processor according to Claim 28, further comprising:

an instruction register coupled to receive instructions from said instruction set, said instruction register operative to hold an instruction to be executed by a data assembly unit; and

a local program memory coupled to said instruction register;

whereby said second functional unit corresponds to said data assembly unit, and said data assembly unit receives an instruction from said second instruction set that causes a separate control thread of instructions to be accessed from said local program memory and executed by said data assembly unit.

43. The embedded-DRAM processor of claim 42, further comprising:

a prefetch unit that prefetches instructions from the first and second instruction sets from a single very long instruction word (VLIW) instruction memory; and

a dispatch unit that dispatches instructions from the first instruction set to the functional units and dispatches instructions from the second instruction stream to the data assembly unit.

44. The embedded-DRAM processor according to Claim 28, wherein said second functional unit monitors execution activity of instructions in said first instruction set and said second instruction set further comprises:

(iv) a command to precharge a row of the DRAM array;

whereby the second functional unit executes a speculative precharging to prevent program delays due to DRAM row precharging.

45. An embedded-DRAM (dynamic random access memory) processor comprising:

an embedded DRAM array comprising a plurality of random access memory cells;

first and second dual-port registers files, whereby the first port of each of said register flies is a parallel access port and is parallely coupled to said DRAM array, each or said register files being capable of being placed into an active state and an inactive state;

a functional unit that executes a first program, said functional unit coupled to said second port of said register files, said functional unit responsive to commands exclusively involving architectural register operands that map onto the registers within a register file that is in the active state;

a data assembly unit responsive to an instruction set comprising at least:

(i) a command that causes data to be moved between the DRAM array and a register file that is in the inactive state; and

(ii) a command that causes said register file in the inactive state to assume the active state and said register file in the active state to assume the inactive state.

46. In a digital processor comprising an embedded DRAM array having a plurality of random access memory cells arranged in rows and columns, at least one row address register, the at least one row address register capable of being loaded or stored, a method comprising:

performing an arithmetic operation on said at least one row address register in order to manipulate a pointer that points to a row of the embedded DRAM array;

speculatively precharging (activating) a row pointed to by said at least one row address register based at least partially upon historical program execution data indicating a possible need to perform one or more load or store operations that would access said row; and

in response to a separate command executed after the speculatively precharging, loading a plurality of words of a row designated by said at least one row address register into designated sets of data registers in a single operation.

47. The method of Claim 46, further comprising deactivating rows pointed to by said at least one row address register.

49. In a digital processor comprising an embedded DRAM array having a plurality of random access memory cells arranged in rows and columns, first and second dual-port registers files each capable of (i) parallel transfer of data between an entire row of said embedded DRAM array in a single operation, and (ii) being placed into an active state and an inactive state, and first and second functional units, a method for processing data comprising:

manipulating data in a data register within a register file that is in an the active state using said first functional unit; and

using said second functional unit;

(a) unidirectionally transferring data between an entire row of said embedded DRAM array and a selected inactive data register file, wherein the transfer of the data occurs in a single operation; and

(b) placing said inactive register file into said active state, whereby when the register file is activated, it becomes an architectural register set of said first functional unit.

50. An embedded-DRAM (dynamic random access memory) processor comprising:

an embedded DRAM array comprising a plurality of random access memory cells arranged in rows and columns;

an embedded row address register that holds a pointer that points to a row of the embedded DRAM array;

embedded first and second registers files, each of said register files capable of loading or storing an entire row of the DRAM array in response to a single latch signal, each of said register files also being capable of being placed into an active state and an inactive state; and

embedded first and second of functional units, said first and second functional units having respective first and second instruction sets and capable of accessing said first and second register files while in said active state;

wherein said first and second registers files comprise a parallel access port operative to parallely transfer contents of said register file between a DRAM row as selected by said row-address register, each of said register file further comprising at least a second access port operative to transfer data between a selected register file and said second functional unit;

wherein said first instruction set comprises at least:

a command to manipulate data in a data register within a register file; and

wherein said second instruction set comprises at least:

a command to perform an arithmetic operation on said row address register; and

a command to load the entire row pointed to by said row address register into a selected set of registers of said register files that is currently in the inactive state, wherein loading the entire row is performed in a single operation.

51. The embedded DRAM processor of claim 1, wherein each of said sets of registers is capable of being loaded or stored in response to a single latch signal without a caching system that employs cache hits and cache misses.

IX. **APPENDIX OF EVIDENCE**

None.

X. **APPENDIX OF RELATED PROCEEDINGS**

None.